Mikrotik - Part4

Profesional

PDF generated using the open source mwlib toolkit. See http://code.pediapress.com/ for more information.
PDF generated at: Thu, 19 Dec 2013 19:19:04 CET

Contents

Articles

Manual:MPLS 1
Manual:MPLS/Overview 4
Manual:MPLS over PPPoE 6
Manual:MPLS/EXP bit behaviour 11
Manual:MPLS/LDP 12
Manual:MPLSVPLS 15
Manual:Cisco VPLS 29
Manual:BGP based VPLS 32
Manual:VPLS Control Word 39
Manual:Virtual Routing and Forwarding 41
Manual:Layer-3 MPLS VPN example 49
Manual:EBGP as PE-CE routing protocol 54
Manual:OSPF as PE-CE routing protocol 60
Manual:TE Tunnels 64
Manual:TE tunnel auto bandwidth 69
Manual:MPLS/Traffic-eng 72
Manual:Interface/Traffic Engineering 75
Manual:Router AAA 78
Manual:PPP AAA 84
Manual:RADIUS Client 89
Manual:User Manager 99
Manual:Hotspot Introduction 102
Manual:Customizing Hotspot 106
Manual:IP/Hotspot 117
Manual:Virtualization 122
Manual:KVM 123
Manual:Xen 132
Manual:Interface/Virtual-ethernet 148

References

Article Sources and Contributors 150
Image Sources, Licenses and Contributors 151

Manual:MPLS

Sub Categories

List of reference sub-pages

- **Interface**
  - vpls
  - traffic-eng
- **MPLS**
  - ldp
  - traffic-eng

Case studies

General

- MPLS Overview and RouterOS MPLS Implementation Status
- EXP bit behaviour
- L2MTU

Layer2 VPN

- LDP and LDP based VPLS
- BGP based VPLS
- Cisco style VPLS
- VPLS Control Word

Layer3 VPN

- Virtual Routing and Forwarding (VRF)
- OSPF as PE-CE routing protocol
- EBGP as PE-CE routing protocol

Traffic Engineering

- TE Tunnels
- TE Tunnel Bandwidth Control

List of examples

General

- MPLS over PPPoE

Layer2 VPN

- P2P L2VPN to Juniper router

Layer3 VPN

- A complete Layer-3 MPLS VPN example
- VRF Route Leaking
- Internet access from VRF
- Internet access from VRF with NAT

Traffic Engineering

- Simple TE configuration
- TE tunnels for VPLS

Summary

MikroTik RouterOS [1] **supports MPLS.** All MikroTik RouterBOARD [2] hardware products support MPLS.

General Porperties

Property	Description
dynamic-label-range (*range of integer[16..1048575]*; Default: **16-1048575**)	Range of Label numbers used for dynamic allocation. First 16 labels are reserved for special purposes (as defined in RFC). If you intend to configure labels statically then adjust dynamic default range not to include numbers that will be used in static configuration.
propagate-ttl (*yes* | *no*; Default: **yes**)	Whether to copy TTL values from IP header to MPLS header. If this option is set to **no** then hops inside MPLS cloud will be invisible from traceroutes.

Forwarding Table

Sub-menu: `/mpls forwarding-table`

Entries in this sub-menu shows label bindings for specific routes that will be used in MPLS label switching. Properties in this menu are read-only

Property	Description
bytes (*integer*)	Total number of packet bytes matched by this entry
destination (*IP/Mask*)	Destination prefix for which labels are assigned
in-label (*integer*)	Label number for incoming packet
interface (*string*)	
ldp (*yes* | *no*)	Whether labels are LDP signaled
nexthop (*IP*)	IP address of the nexthop
out-label (*integer*)	Label number which is added or switched to for outgoing packet.
packets (*integer*)	Number of packets matched by this entry
traffic-eng (*yes* | *no*)	Shows whether entry is signaled by RSVP-TE (Traffic Engineering)
vpls (*yes* | *no*)	Shows whether entry is used for VPLS tunnels.

For example we have forwarding table as shown below.

```
[admin@RB493G] /mpls forwarding-table> print
Flags: L - ldp, V - vpls, T - traffic-eng
 #    IN-LABEL      OUT-LABELS    DESTINATION                    IN NEXTHOP
 0    expl-null
 1 L  105                         10.255.255.36/32               lo 10.5.101.36
 2 L  120           112           3.3.3.1/32                     lo 10.5.101.3
 3 L  121           113           3.3.3.2/32                     lo 10.5.101.3
[admin@RB493G] /mpls forwarding-table>
```

You can see that all labels are LDP signaled. Note that for entry #1 there is no out-label, it means that MPLS label switching will not occur, packet will be sent out as regular IP packet. In the other hand entry #2 has in-label and out-label, which means that during packet forwarding label will be switched from 120 to 112.

Interface

Sub-menu: `/mpls interface`

This menu allows to configure MTUs including MPLS headers that interface can forward without fragmentation.

Note: If Ethernet card does not support Jumbo frames, then MPLS MTU for all interfaces on all devices participating in LSP should be set to 1500

Properties

Property	Description
comment (*string*; Default:)	Short description of the interface
disabled (*yes* | *no*; Default: **no**)	If set to **yes** then this configuration is ignored.
interface (*string* | *all*; Default: **all**)	Interface name to which apply settings. If set to **all** then the same config will be used for every interface if there is no specific configuration for the interface.
mpls-mtu (*integer [512..65535]*; Default: **1508**)	Option represents how big packets can be carried over the interface with added MPLS labels. Read More >>

In RouterOS by default have entry which sets MS MTU to 1508 for all interfaces.

```
[admin@RB493G] /mpls interface> print
Flags: X - disabled
 #   INTERFACE                                                       MPLS-MTU
 0   all                                                                 1508
```

Local Bindings

Sub-menu: /mpls local-bindings

This sub-menu shows labels bound to the routes locally in the router. In this menu also static bindings can be configured if there is no intention to use any of dynamic protocols (like LDP).

Properties

Property	Description
comments (*string*; Default:)	Short description of the entry
disabled (*yes* | *no*; Default: **no**)	
dst-address (*IP/Mask*; Default:)	Destination prefix for which label is assigned
label (*integer[0..1048576]* | *alert* | *expl-null* | *expl-null6* | *impl-null* | *none*; Default:)	Label number assigned to destination.

Read-only Properties

Property	Description
adv-path ()	
advertised (*yes* | *no*)	Whether binding was advertised to the neigbors
dynamic (*yes* | *no*)	Whether entry was dynamically added
egress (*yes* | *no*)	
gateway-route (*yes* | *no*)	Whether destination is reachable through the gateway.
local-route (*yes* | *no*)	Whether destination is locally reachable on the router
peers (*IP:label_space*)	IP address and label space of the peer to which this entry was advertised.

Remote Bindings

Sub-menu: `/mpls remote-bindings`

Sub-menu shows label bindings for routes received from other routers. This table is used to build Forwarding Table

[Top | Back to Content **]**

References

[1] http://mikrotik.com/software.html
[2] http://routerboard.com

Manual:MPLS/Overview

MPLS Overview

MPLS stands for MultiProtocol Label Switching. It kind of replaces IP routing - packet forwarding decision (outgoing interface and next hop router) is no longer based on fields in IP header (usually destination address) and routing table, but on labels that are attached to packet. This approach speeds up forwarding process because next hop lookup becomes very simple compared to routing lookup (finding longest matching prefix).

Efficiency of forwarding process is the main benefit of MPLS, but it must be taken into account that MPLS forwarding disables processing of network layer (e.g. IP) headers, therefore no network layer based actions like NAT and filtering can be applied to MPLS forwarded packets. Any network layer based actions should be taken on ingress or egress of MPLS cloud, with preferred way being ingress - this way, e.g. traffic that is going to be dropped anyway does not travel through MPLS backbone.

In the simplest form MPLS can be thought of like improved routing - labels are distributed by means of LDP protocol for routes that are active and labeled packet takes the same path it would take if it was not labeled. Router that routes unlabeled packet using some route for which it has received label from next hop, imposes label on packet and send it to next hop - it gets MPLS switched further along its path. Router that receives packet with label it has assigned to some route changes packet label with one received from next hop of particular route and sends packet to next hop. Label switched path ensures delivery of data to the MPLS cloud egress point. Applications of MPLS are based on this basic MPLS concept of label switched paths.

Another way of establishing label switching path is traffic engineering tunnels (TE tunnels) by means of RSVP TE protocol. Traffic engineering tunnels allow explicitly routed LSPs and constraint based path selection (where constraints are interface properties and available bandwidth).

Taking into account complexity, new protocols and applications that MPLS introduces and differences of concepts that MPLS adds to routed/bridged network, it is recommended to have in depth understanding of MPLS concepts before implementing MPLS in production network. Some suggested reading material:

- Multiprotocol Label Switching http://en.wikipedia.org/wiki/Multiprotocol_Label_Switching
- RFC3031 Multiprotocol Label Switching Architecture http://www.ietf.org/rfc/rfc3031.txt
- MPLS Fundamentals by Luc De Ghein http://www.amazon.com/MPLS-Fundamentals-Luc-Ghein/dp/1587051974

RouterOS MPLS features

As of version 3.8 MPLS feature development for RouterOS continues in **mpls-test** package that requires **routing-test** package. Currently RouterOS (by means of mpls-test and routing-test packages) supports the following MPLS related features:

- MPLS switching with penultimate hop popping support
- static local label bindings for IPv4
- static remote label bindings for IPv4
- Label Distribution Protocol (RFC 3036, RFC 5036) for IPv4
  - downstream unsolicited label advertisement
  - independent label distribution control
  - liberal label retention
  - targeted session establishment
  - optional loop detection
- Virtual Private Lan Service
  - VPLS LDP signaling (RFC 4762)
  - VPLS pseudowire fragmentation and reassembly (RFC 4623)
  - VPLS MP-BGP based autodiscovery and signaling (RFC 4761), see BGP based VPLS
- RSVP TE Tunnels
  - tunnel head-end
  - explicit paths
  - OSPF extensions for TE tunnels
  - CSPF path selection
  - forwarding of VPLS and MPLS IP VPN traffic on TE tunnels
- MP-BGP based MPLS IP VPN
- OSPF extensions for MPLS TE

Features since version 3.17:

- support for OSPF as CE-PE protocol
- ping and traceroute for specified VRF
- control over network layer TTL propagation in MPLS

Features since version 3.20 (note that this version changes configuration syntax and adds new parameters!):

- Cisco style static VPLS pseudowires (RFC 4447 FEC type 0x80)
- Cisco VPLS BGP-based auto-discovery (draft-ietf-l2vpn-signaling-08)
- support for multiple import/export route target extended communities for BGP based VPLS (both, RFC 4761 and draft-ietf-l2vpn-signaling-08)

Features since version 3.23

- Ingress TE tunnel rate limit and automatic reserved bandwidth adjustment, see TE Tunnel Bandwidth Control
- all tunnel bandwidth settings are specified and displayed in bits per second
- complete support for OSPF as PE-CE routing protocol (including sham links)

Features since version 3.24

- RIP as CE-PE protocol
- per-VRF BGP instance redistribution settings

MPLS features that RouterOS DOES NOT HAVE yet:

- IPv6 support
- LDP features:

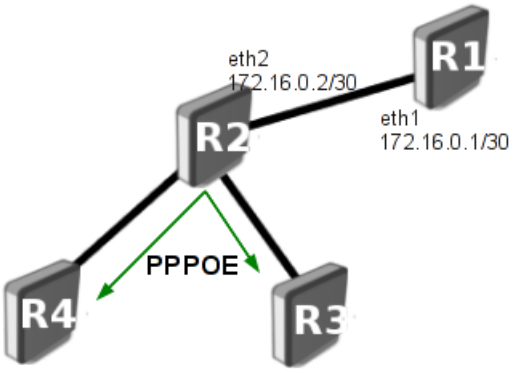
  - downstream on demand label advertisement
  - ordered label distribution control
  - conservative label retention
- TE features
  - fast reroute
  - link/node protection
- Support for BGP as label distribution protocol

To ensure compatibility with other manufacturer equipment ensure that required features match, if uncertain, consult with Mikrotik support. RouterOS LDP and TE implementation has been tested with Cisco IOS.

Manual:MPLS over PPPoE

Applies to RouterOS: v3, v4, v5

This example shows how to set up MPLS network over PPPoE interfaces.



As you ca see from illustration above, router R2 is pppoe server and routers R3 and R4 are pppoe clients. Our goal is to run MPLS on this network.

When running MPLS over PPPoE or other tunnels you have to deal with MTU issues. Tunnels add more overhead (in our case PPPoE adds 8 more bytes). To be able to forward 1500 byte IP packet without fragmentation we will need interface that supports

1500 (IP frame)

+ 8 (PPPoE header)

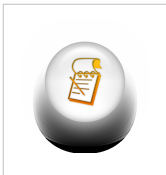
+ 4 (MPLS header)

= 1512bytes

From RouterBoard MTU table you can check if RouterBoard supports 1512 L2MTU.

Lets say that R2 is RB433 and pppoe clients are connected to ether2. From the table you can see that max supported l2MTU for this interface is 1522.

It means that router will be able to forward packets without fragmentations.



Note: Since v5.0 is added proper support for MPLS over PPP. Now by default MPLS is disabled, to enable it go to
`/ppp profile` menu and set `use-mpls=`**`yes`**

```
/system identity set name=R1

# add loopback interface
/interface bridge
add name=loopback
/ip address
add address=10.255.255.1/32 interface=loopback
add address=172.16.0.1/30 interface=ether1

#set up ospf
/routing ospf instance
set default redistribute-connected=as-type-1
/routing ospf network
add network=172.16.0.0/30 area=backbone

# set up MPLS/LDP
/mpls interface set 0 mpls-mtu=1512
/mpls ldp
set enabled=yes lsr-id=10.255.255.1 transport-address=10.255.255.1
/mpls ldp interface
add interface=ether1
```

Note that we have to add static interface for each PPPoE clients, because later on these interfaces will be added to LDP configuration.

```
/system identity set name=R2

# add loopback interface
/interface bridge
add name=loopback
/ip address
add address=10.255.255.2/32 interface=loopback
add address=172.16.0.2/30 interface=ether1

# set up pppoe
/interface pppoe-server server
add interface=ether2 service-name=mpls max-mru=1500 max-mtu=1500

/ppp secret
add name=mplsR3 service=pppoe remote-address=192.168.0.2 local-address=192.168.0.1
add name=mplsR4 service=pppoe remote-address=192.168.0.3 local-address=192.168.0.1
```

```
/interface pppoe-server
add name="mplsR3" user="mplsR3" service="mpls"
add name="mplsR4" user="mplsR4" service="mpls"

#set up ospf
/routing ospf instance
set default redistribute-connected=as-type-1
/routing ospf network
add network=172.16.0.0/30 area=backbone
add network=192.168.0.2/32 area=backbone
add network=192.168.0.3/32 area=backbone

# set up MPLS/LDP
/mpls interface set 0 mpls-mtu=1512
/mpls ldp
set enabled=yes lsr-id=10.255.255.2 transport-address=10.255.255.2
/mpls ldp interface
add interface=ether1
add interface=mplsR3
add interface=mplsR4
```

```
/system identity set name=R3

# add loopback interface
/interface bridge
add name=loopback
/ip address
add address=10.255.255.3/32 interface=loopback

# set up pppoe
/interface pppoe-client
add name="mplsR3" max-mtu=1500max-mru=1500 interface=ether2 user="mplsR3" service-name=mpls

#set up ospf
/routing ospf instance
set default redistribute-connected=as-type-1
/routing ospf network
add network=192.168.0.1/32 area=backbone

# set up MPLS/LDP
/mpls interface set 0 mpls-mtu=1512
/mpls ldp
set enabled=yes lsr-id=10.255.255.3 transport-address=10.255.255.3
/mpls ldp interface
add interface=mplsR3
```

```
/system identity set name=R4
```

```
# add loopback interface
/interface bridge
add name=loopback
/ip address
add address=10.255.255.4/32 interface=loopback

# set up pppoe
/interface pppoe-client
add name="mplsR4" max-mtu=1500 max-mru=1500 interface=ether2 user="mplsR4" service-name=mpls

#set up ospf
/routing ospf instance
set default redistribute-connected=as-type-1
/routing ospf network
add network=192.168.0.1/32 area=backbone

# set up MPLS/LDP
/mpls interface set 0 mpls-mtu=1512
/mpls ldp
set enabled=yes lsr-id=10.255.255.4 transport-address=10.255.255.4
/mpls ldp interface
add interface=mplsR4
```

At first make sure pppoe clients are connected successfully

```
[admin@R2] /ppp active> print
Flags: R - radius
 #   NAME          SERVICE CALLER-ID          ADDRESS         UPTIME   ENCODING
 0   mplsR3        pppoe   00:0C:42:21:F1:EA 192.168.0.2     46m
 1   mplsR4        pppoe   00:0C:42:21:F1:ED 192.168.0.3     46m55s
```

Check if OSPF is running properly

```
[admin@R2] /routing ospf neighbor> print
 0 router-id=10.255.255.1 address=172.16.0.1 interface=wlan1 priority=1
   dr-address=172.16.0.2 backup-dr-address=172.16.0.1 state="Full"
   state-changes=5 ls-retransmits=0 ls-requests=0 db-summaries=0
   adjacency=5m19s

 1 router-id=10.255.255.3 address=192.168.0.2 interface=mplsR3 priority=1
   dr-address=0.0.0.0 backup-dr-address=0.0.0.0 state="Full" state-changes=4
   ls-retransmits=0 ls-requests=0 db-summaries=0 adjacency=49m33s

 2 router-id=10.255.255.4 address=192.168.0.3 interface=mplsR4 priority=1
   dr-address=0.0.0.0 backup-dr-address=0.0.0.0 state="Full" state-changes=4
   ls-retransmits=0 ls-requests=0 db-summaries=0 adjacency=50m31s
```

Ensure LDP is running

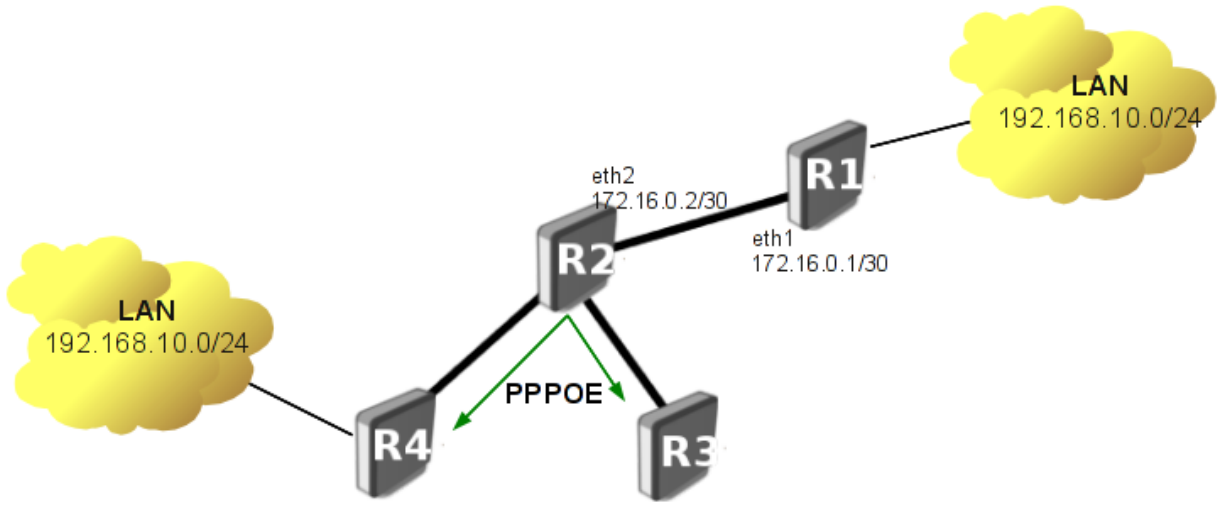
```
[admin@R2] /mpls ldp neighbor> print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello,
V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEN
 0 DO   10.255.255.3    10.255.255.2    10.255.255.3:0            no
 1 DO   10.255.255.4    10.255.255.2    10.255.255.4:0            no
 2 DO   10.255.255.1    10.255.255.2    10.255.255.1:0            no
```

```
[admin@R2] /mpls forwarding-table> print
Flags: L - ldp, V - vpls, T - traffic-eng
 #    IN-LABEL      OUT-LABELS  DESTINATION                   I NEXTHOP
 0    expl-null
 1 L  20                        192.168.0.1/32                m 192.168.0.3
 2 L  21                        10.255.255.4/32               m 192.168.0.3
 3 L  22                        10.255.255.3/32               m 192.168.0.2
 4 L  23                        10.255.255.1/32               w 172.16.0.1
 5 L  24                        192.168.88.0/24               w 172.16.0.1
```

Now we can check if packet switching is working as expected

```
[admin@R4] /mpls ldp neighbor> /tool traceroute 10.255.255.1 src-address=10.255.255.4
     ADDRESS                                  STATUS
   1     192.168.0.1 13ms 19ms 143ms
                      mpls-label=23
   2    10.255.255.1 38ms 15ms 14ms
```

This example extends previous setup by connecting two local networks using VPLS tunnel



Manual:MPLS/EXP bit behaviour

MPLS label EXP field overview

When MPLS label is attached to packet, it increases packet length by 32 bits (4 bytes). These 32 bits are broken down as follows:

- label value itself (20 bits)
- EXP ("experimental") field (3 bits)
- time to live field (8 bits)
- bottom of stack field (1 bit)

Use of "experimental" bits is not specified by MPLS standards, but most common use is to carry QoS information, similar to 802.1q priority in VLAN tag. Note that EXP field is 3 bits only therefore it can carry values from 0 to 7 only, which allows to have 8 traffic classes.

EXP field treatment in RouterOS

When RouterOS receives MPLS packet, it sets "ingress priority" value for packet to that carried inside top label. Note that "ingress priority" **is not** a field inside packet headers - it can be thought of like additional mark assigned to packet while being processed by router. When RouterOS labels MPLS packet, it sets EXP bits to "priority" (not "ingress priority"!) assigned to packet. When RouterOS switches MPLS packet, "ingress priority" is automatically copied to "priority", this way regular MPLS switching communicates priority info over whole label switched path.

Additional info on "ingress priority" and "priority" handling is also in WMM.

Therefore what happens to EXP field depends based on what action is taken on packet:

- if packet is MPLS switched (by popping label off packet and pushing on new one), EXP field in new label will be the same as in received label, because:
  - RouterOS sets "ingress priority" to EXP bits in received label
  - Switching automatically sets "priority" to "ingress priority"
  - RouterOS labels packet with new label and sets its EXP bits to value in "priority".
- if packet is MPLS switched by using penultimate-hop-popping (received label is popped off and no new one is pushed on), EXP field of received priority stays in "priority" field of packet and may be used by some other MAC protocol, e.g. WMM or 802.1q VLAN, for example:
  - RouterOS sets "ingress priority" to EXP bits in received label
  - Switching automatically sets "priority" to "ingress priority"
  - RouterOS switches packet to next hop (without pushing on label) and that happens over VLAN interface
  - VLAN interface sets 802.1q priority in VLAN header to "priority" value of packet.

Note that penultimate-hop-popping can therefore loose QoS information carried over label switched path at the last hop. In cases where this is not desirable, penultimate-hop-popping behaviour should be disabled by using Explicit NULL label instead of Implicit NULL label for last hop in label switched path. Using Explicit NULL label for last hop is default behaviour for MPLS TE tunnels.

- if packet is supposed to be sent over label switched path (first label will get pushed on packet), EXP bits will be set to value in "priority", which in turn can be set up properly using firewall rules or other means (e.g. from DSCP field in IP header)
- if packet is received for local processing, "ingress priority" is set to EXP field of received packet and can therefore be used to update DSCP field of packet or set "priority" from "ingress priority" using firewall rules

Manual:MPLS/LDP

Applies to RouterOS: v3, v4 +

Summary

MikroTik RouterOS implements Label Distribution Protocol (RFC 3036, RFC 5036) for IPv4. LDP is a protocol defined for distributing labels. It is the set of procedures and messages by which Label Switched Routers (LSRs) establish Label Switched Paths (LSPs) through a network by mapping network-layer routing information directly to data-link layer switched paths.

General

Sub-menu: `/mpls ldp`

General LDP settings.

Properties

Property	Description
distribute-for-default-route (*yes* | *no*; Default: **no**)	Defines whether to distribute label for default route.
enabled (*yes* | *no*; Default: **yes**)	Disable or enable LDP protocol.
hoplimit (*integer [0..4294967295]*; Default: **255**)	Max hop limit used for loop detection. Works in combination with **loop-detect** property.
loop-detect (*yes* | *no*; Default: **no**)	Defines whether to run LSP loop detection. Will not work correctly if not enabled on all LSRs. Should be used only on non-TTL networks such as ATM.
lsr-id (*IP*; Default: **0.0.0.0**)	Unique label switching router's ID. If set to **0.0.0.0** highest IP address on the router is used.
path-vector-limit (*integer[0..4294967295]*; Default: **255**)	Max path vector limit used for loop detection. Works in combination with **loop-detect** property.
transport-address (*IP*; Default: **0.0.0.0**)	Specifies LDP session connections origin address and also advertise this address as transport address to LDP neighbors. If set to **0.0.0.0** highest IP address on the router is used.
use-explicit-null (*yes* | *no*>; Default: **no**)	Whether to distribute explicit-null label bindings.

Interface

Sub-menu: `/mpls ldp interface`

List of interfaces that connects Label Switching routers.

Properties:

Property	Description
accept-dynamic-neighbors (*yes* | *no*; Default: **yes**)	Defines whether to discover neighbors dynamically or use only statically configured in LDP neighbors menu
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used
hello-interval (*time*; Default: **5s**)	The interval between hello packets that the router sends out this interface.
hold-time (*time*; Default: **15s**)	Specifies the interval after which a neighbor is declared as not reachable.
transport-address (*IP*; Default: **0.0.0.0**)	Used transport address if differs from general settings. If set to **0.0.0.0** transport address from general settings is used.
interface (*string*; Default:)	Name of the interface on which to run LDP

Neighbors

Sub-menu: `/mpls ldp neighbor`

Properties

Property	Description
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used
send-targeted (*yes* | *no*; Default: **yes**)	Specifies whether to send targeted hellos, used for targeted (not directly connected) LDP sessions.
transport (*IP*; Default:)	Transport address used by remote neighbor.

Read-only properties

Property	Description
addresses (*IP[,IP[,...]]*)	List of all IP addresses on LDP neighbor
dynamic (*yes* | *no*)	Shows whether item is dynamically created
local-transport (*IP*)	Transport address used to send messages to the neighbor.
operational (*yes* | *no*)	Shows whether item is running
peer (*LSR-ID:integer*)	Shows remote neighbor's LSR ID and label space.
sending-targeted-hello (*yes* | *no*)	Shows whether targeted hellos are sent to the neighbor.
vpls (*yes* | *no*)	Shows whether the neighbor is used for VPLS tunnel.

Accept Filters

Sub-menu: `/mpls ldp accept-filter`

List of label bindings which should be accepted from LDP neighbors.

Properties:

Property	Description
accept (*yes* | *no*; Default: **yes**)	Whether to accept label bindings from the neighbors for specified prefix.
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used
neighbor (*all* | *IP*; Default: **all**)	Neighbor to which this filter applies.
prefix (*IP/Mask*; Default: **0.0.0.0/0**)	

Advertise Filters

Sub-menu: `/mpls ldp advertise-filter`

List of label bindings which should be advertised to LDP neighbors.

Properties:

Property	Description
accept (*yes* | *no*; Default: **yes**)	Whether to advertise label bindings to the neighbors for specified prefix.
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used
neighbor (*all* | *IP*; Default: **all**)	Neighbor to which this filter applies.
prefix (*IP/Mask*; Default: **0.0.0.0/0**)	

[Top | Back to Content]

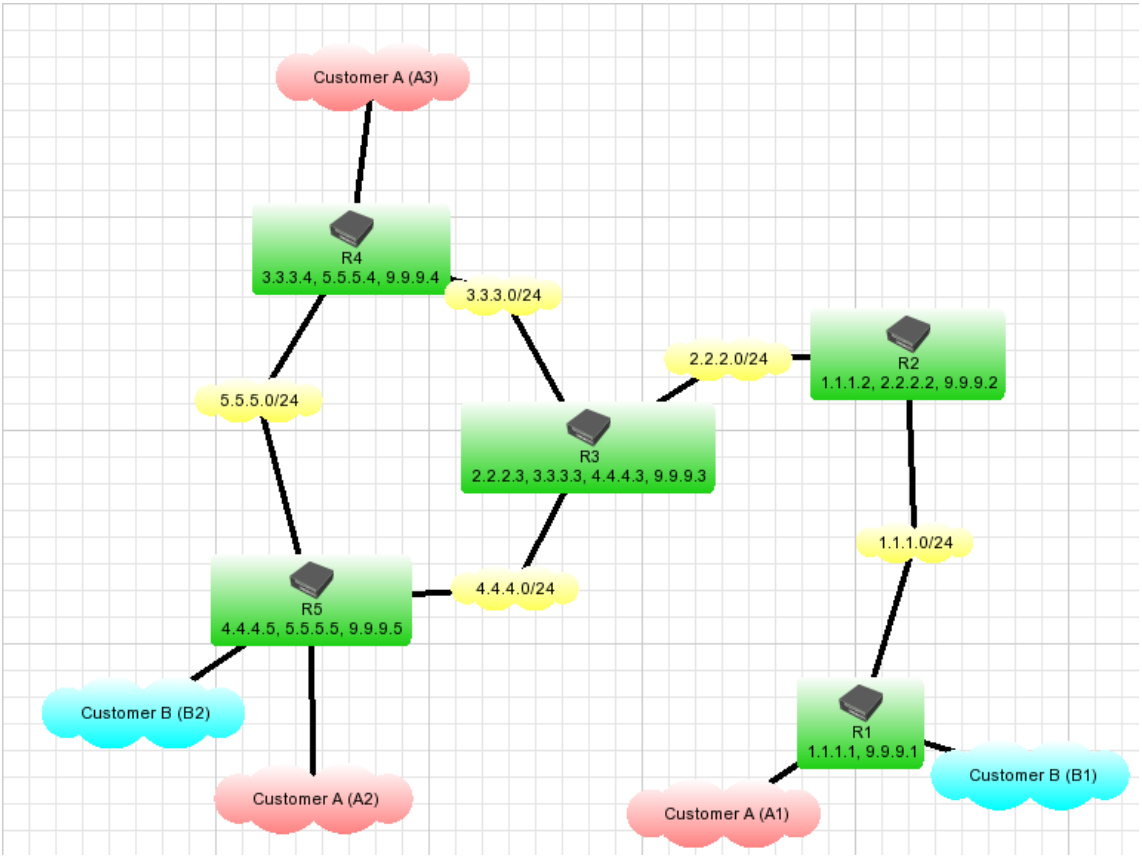
Manual:MPLSVPLS

MPLS Overview

For MPLS overview and MPLS features that RouterOS supports see MPLS Overview

Example network

Consider network service provider that is connecting 3 remote sites of Customer A (A1,A2 and A3) and 2 remote sites of Customer B (B1 and B2) using its routed IP network core, consisting of routers (R1-R5):



Customers require transparent ethernet segment connection between sites. So far it has been implemented by means of bridging EoIP tunnels with physical ethernet interfaces.

Note that there are no IP addresses configured on R1, R4 and R5 interfaces that face customer networks.

Enabling MPLS forwarding can speed up packet forwarding process in such network. Using one of MPLS applications - VPLS can further increase efficency of ethernet frame forwarding by not having to encapsulate ethernet frames in IP frames, thus removing IP header overhead.

This guide gives step by step instructions that will lead to implementation of VPLS to achieve necessary service.

Prerequisites for MPLS

"Loopback" IP address

Although not a strict requirement, it is advisable to configure routers participating in MPLS network with "loopback" IP addresses (not attached to any real network interface) to be used by LDP to establish sessions.

This serves 2 purposes:

- as there is only one LDP session between any 2 routers, no matter how many links connect them, loopback IP address ensures that LDP session is not affected by interface state or address changes
- use of loopback address as LDP transport address ensures proper penultimate hop popping behaviour when multiple labels are attached to packet as in case of VPLS

In RouterOS "loopback" IP address can be configured by creating dummy bridge interface without any ports and adding address to it. For example, on R1 it is done with the following commands:

```
/interface bridge add name=lobridge
/ip address add address=9.9.9.1/32 interface=lobridge
```

The rest of routers are configured similar way.

IP connectivity

As LDP distributes labels for active routes, essential requirement is properly configured IP routing. LDP by default distributes labels for active IGP routes (that is - connected, static, and routing protocol learned routes, except BGP).

In given example setup OSPF is used to distribute routes. For example, on R5 OSPF is configured with the following commands:

```
/routing ospf set redistribute-connected=as-type-1
/routing ospf network add area=backbone network=4.4.4.0/24
/routing ospf network add area=backbone network=5.5.5.0/24
```

On other routers OSPF is configured in similar way.

This yields routing table on R5 like this:

```
[admin@R5] > ip route print
Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY-STATE GATEWAY         DISTANCE             INTERFACE
 0 ADo  1.1.1.0/24                         reachable     4.4.4.3         110                  ether1
 1 ADo  2.2.2.0/24                         reachable     4.4.4.3         110                  ether1
 2 ADo  3.3.3.0/24                         reachable     4.4.4.3         110                  ether1
                                           reachable     5.5.5.4                              ether2
 3 ADC  4.4.4.0/24         4.4.4.5                                       0                    ether1
 4 ADC  5.5.5.0/24         5.5.5.5                                       0                    ether2
 5 ADo  9.9.9.1/32                         reachable     4.4.4.3         110                  ether1
 6 ADo  9.9.9.2/32                         reachable     4.4.4.3         110                  ether1
 7 ADo  9.9.9.3/32                         reachable     4.4.4.3         110                  ether1
 8 ADo  9.9.9.4/32                         reachable     5.5.5.4         110                  ether2
 9 ADC  9.9.9.5/32         9.9.9.5                                       0                    lobridge
```

and traceroute from R5 to R1 like this:

```
[admin@R5] > /tool traceroute 9.9.9.1
     ADDRESS                                    STATUS
   1          4.4.4.3 11ms 1ms 4ms
   2          2.2.2.2 23ms 3ms 2ms
   3          9.9.9.1 26ms 3ms 3ms
```

Configuring LDP

In order to distribute labels for routes, LDP should get enabled. On R1 this is done by commands (interface ether3 is facing network 1.1.1.0/24):

```
/mpls ldp set enabled=yes transport-address=9.9.9.1 lsr-id=9.9.9.1
/mpls ldp interface add interface=ether3
```

Note that transport-address gets set to 9.9.9.1. This makes router originate LDP session connections with this address and also advertise this address as transport address to LDP neighbors.

Other routers are configured in similar way - LDP is enabled on interfaces that connect routers and not enabled on interfaces that connect customer networks. For example, on R5:

```
[admin@R5] > /ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   4.4.4.5/24         4.4.4.0         4.4.4.255       ether1
 1   5.5.5.5/24         5.5.5.0         5.5.5.255       ether2
 2   9.9.9.5/32         9.9.9.5         9.9.9.5         lobridge
[admin@R5] > /mpls ldp interface print
Flags: I - invalid, X - disabled
 #   INTERFACE                                      HELLO-INTERVAL       HOLD-TIME
 0   ether1                                         5s                   15s
 1   ether2                                         5s                   15s
```

After LDP sessions are established, on R5 there are 2 LDP neighbors:

```
[admin@R5] > /mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                          SEND-TARGETED ADDRESSES
 0 DO   9.9.9.4         9.9.9.5         9.9.9.4:0                     no            3.3.3.4
                                                                                      5.5.5.4
                                                                                      9.9.9.4
 1 DO   9.9.9.3         9.9.9.5         9.9.9.3:0                     no            2.2.2.3
                                                                                      3.3.3.3
                                                                                      4.4.4.3
                                                                                      9.9.9.3
```

/mpls local-bindings shows labels that this router has assigned to routes and peers it has distributed the label to. It shows that R5 has distributed labels for all its routes to both of its neighbors - R3 and R4

```
[admin@R5] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 4.4.4.0/24         impl-null  9.9.9.4:0
```

```
                                    9.9.9.3:0
 1 ADLe 9.9.9.5/32         impl-null  9.9.9.4:0
                                    9.9.9.3:0
 2 ADG  9.9.9.4/32         17         9.9.9.4:0
                                    9.9.9.3:0
 3 ADLe 5.5.5.0/24         impl-null  9.9.9.4:0
                                    9.9.9.3:0
 4 ADG  1.1.1.0/24         18         9.9.9.4:0
                                    9.9.9.3:0
 5 ADG  2.2.2.0/24         19         9.9.9.4:0
                                    9.9.9.3:0
 6 ADG  9.9.9.1/32         20         9.9.9.4:0
                                    9.9.9.3:0
 7 ADG  9.9.9.2/32         21         9.9.9.4:0
                                    9.9.9.3:0
 8 ADG  9.9.9.3/32         22         9.9.9.4:0
                                    9.9.9.3:0
 9 ADG  3.3.3.0/24         23         9.9.9.4:0
                                    9.9.9.3:0
```

/mpls remote-bindings shows labels that are allocated for routes by neighboring routers and advertised to this router:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP          LABEL      PEER
 0  D 4.4.4.0/24                          16         9.9.9.4:0
 1 AD 3.3.3.0/24         5.5.5.4          impl-null  9.9.9.4:0
 2  D 9.9.9.5/32                          17         9.9.9.4:0
 3 AD 9.9.9.4/32         5.5.5.4          impl-null  9.9.9.4:0
 4  D 5.5.5.0/24                          impl-null  9.9.9.4:0
 5  D 1.1.1.0/24                          18         9.9.9.4:0
 6  D 2.2.2.0/24                          19         9.9.9.4:0
 7  D 9.9.9.1/32                          20         9.9.9.4:0
 8  D 9.9.9.2/32                          21         9.9.9.4:0
 9  D 9.9.9.3/32                          22         9.9.9.4:0
10 AD 1.1.1.0/24         4.4.4.3          16         9.9.9.3:0
11 AD 2.2.2.0/24         4.4.4.3          impl-null  9.9.9.3:0
12  D 4.4.4.0/24                          impl-null  9.9.9.3:0
13  D 3.3.3.0/24                          impl-null  9.9.9.3:0
14 AD 9.9.9.1/32         4.4.4.3          17         9.9.9.3:0
15 AD 9.9.9.3/32         4.4.4.3          impl-null  9.9.9.3:0
16  D 9.9.9.4/32                          18         9.9.9.3:0
17  D 5.5.5.0/24                          19         9.9.9.3:0
18 AD 9.9.9.2/32         4.4.4.3          20         9.9.9.3:0
19  D 9.9.9.5/32                          21         9.9.9.3:0
```

Here we can observe that R5 has received label bindings for all routes from both its neighbors - R3 and R4, but only the ones for whom particular neighbor is next hop are active. For example:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE             INTERFACE
...
 5 ADo  9.9.9.1/32                         r 4.4.4.3         110                  ether1
...
```

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS         NEXTHOP          LABEL      PEER
...
 7  D 9.9.9.1/32                          20         9.9.9.4:0
...
14 AD 9.9.9.1/32          4.4.4.3          17         9.9.9.3:0
...
```

From the above we see that R3, which is next hop for network 9.9.9.1/32 from R5 perspective, has assigned label 17 for traffic going to 9.9.9.1/32. This implies that when R5 will be routing traffic to this network, will impose label 17.

Label switching rules can be seen in /mpls forwarding-table. For example, on R3 it looks like this:

```
[admin@R3] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE            NEXTHOP
 ...
 2 17                   17                   9.9.9.1/32         ether1               2.2.2.2
 ...
```

This rule says that R3 receiving packet with label 17 will change it to label 17 assigned by R2 for network 9.9.9.1/32 (R2 is next hop for 9.9.9.1/32 from R3 perspective):

```
[admin@R2] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 ...
 3 ADG  9.9.9.1/32         17         9.9.9.1:0
                                      9.9.9.3:0
 ...
```

R2 MPLS forwarding table tells:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE            NEXTHOP
 ...
 1 17                                        9.9.9.1/32         ether1               1.1.1.1
 ...
```

Notice, that forwarding rule does not have any out-labels. The reason for this is that R2 is doing penultimate hop popping for this network. R1 does not assign any real label for 9.9.9.1/32 network, because it is known that R1 is egress point for 9.9.9.1/32 network (router is egress point for networks that are directly connected to it, because next hop for traffic is not MPLS router), therefore is advertises "implicit null" label for this route:

```
[admin@R2] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
13 AD 9.9.9.1/32         1.1.1.1         impl-null  9.9.9.1:0
 ...
```

This tells R2 to forward traffic for 9.9.9.1/32 to R1 unlabelled, which is exactly what R2 mpls forwarding-table entry tells. Penultimate hop popping ensures that routers do not have to do unnecessary label lookup when it is known in advance that router will have to route packet.

Using traceroute in MPLS networks

RFC4950 introduces extensions to ICMP protocol for MPLS. The basic idea is that some ICMP messages may carry MPLS "label stack object" (list of labels that were on packet when it caused particular ICMP message). ICMP messages of interest for MPLS are Time Exceeded and Need Fragment.

MPLS label carries not only label value, but also TTL field. When imposing label on IP packet, MPLS TTL is set to value in IP header, when last label is removed from IP packet, IP TTL is set to value in MPLS TTL. Therefore MPLS switching network can be diagnosed by means of traceroute tool that supports MPLS extension.

For example, traceroute from R5 to R1 looks like this:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                    STATUS
   1         4.4.4.3 15ms 5ms 5ms
                      mpls-label=17
   2         2.2.2.2 5ms 3ms 6ms
                      mpls-label=17
   3         9.9.9.1 6ms 3ms 3ms
```

Traceroute results show MPLS labels on packet when it produced ICMP Time Exceeded. The above means: when R3 received packet with MPLS TTL 1, it had label 17 on. This matches label advertised by R3 for 9.9.9.1/32. The same way R2 observed label 17 on packet on next traceroute iteration - R3 switched label 17 to label 17, as explaned above. R1 received packet without labels - R2 did penultimate hop popping as explaned above.

Drawbacks of using traceroute in MPLS network

Label switching ICMP errors

One of drawbacks of using traceroute in MPLS networks is the way MPLS handles produced ICMP errors. In IP networks ICMP errors are simply routed back to source of packet that caused the error. In MPLS network it is possible that router that produces error message does not even have route to source of IP packet (for example in case of assymetric label switching paths or some kind of MPLS tunneling, e.g. to transport MPLS VPN traffic).

Due to this produced ICMP errors are not routed to the source of packet that caused the error, but switched further along label switching path, assuming that when label switching path endpoint will receive ICMP error, it will know how to properly route it back to source.

This causes the situation, that traceroute in MPLS network can not be used the same way as in IP network - to determine failure point in the network. If label switched path is broken anywhere in the middle, no ICMP replies will come back, because they will not make it to the far endpoint of label switching path.

Penultimate hop popping and traceroute source address

Thorough understanding of pen ultimate hop behaviour and routing is necessary to understand and avoid problems that penultimate hop popping causes to traceroute.

In the example setup, regular traceroute from R5 to R1 would yield the following results:

```
[admin@R5] > /tool traceroute 9.9.9.1
     ADDRESS                                  STATUS
   1         0.0.0.0 timeout timeout timeout
   2         2.2.2.2 37ms 4ms 4ms
                      mpls-label=17
   3         9.9.9.1 4ms 2ms 11ms
```

compared to:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                  STATUS
   1         4.4.4.3 15ms 5ms 5ms
                      mpls-label=17
   2         2.2.2.2 5ms 3ms 6ms
                      mpls-label=17
   3         9.9.9.1 6ms 3ms 3ms
```

The reason why first traceroute does not get response from R3 is that by default traceroute on R5 uses source address 4.4.4.5 for its probes, because it is preferred source for route over which next hop to 9.9.9.1/32 is reachable:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE             INTERFACE
 ...
 3 ADC  4.4.4.0/24         4.4.4.5                           0                    ether1
 ...
 5 ADo  9.9.9.1/32                         r 4.4.4.3         110                  ether1
 ...
```

When first traceroute probe is transmitted (source: 4.4.4.5, destination 9.9.9.1), R3 drops it and produces ICMP error message (source 4.4.4.3 destination 4.4.4.5) that is switched all the way to R1. R1 then sends ICMP error back - it gets switched along label switching path to 4.4.4.5.

R2 is penultimate hop popping router for network 4.4.4.0/24, because 4.4.4.0/24 is directly connected to R3. Therefore R2 removes last label and sends ICMP error to R3 unlabelled:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE            NEXTHOP
 ...
 3 19                                        4.4.4.0/24         ether2               2.2.2.3
 ...
```

R3 drops received IP packet, because it receives packet with its own address as source address. ICMP errors produced by following probes come back correctly, because R3 receives unlabelled packets with source addresses 2.2.2.2 and 9.9.9.1, which are acceptable to route.

Command:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
 ...
```

produces expected results, because source address of traceroute probes is 9.9.9.5. When ICMP errors are travelling back from R1 to R5, penultimate hop popping for 9.9.9.5/32 network happens at R3, therefore it never gets to route packet with its own address as source address.

Configuring VPLS

Configuring VPLS interfaces

VPLS interface can be considered tunnel interface just like EoIP interface. To achieve transparent ethernet segment forwarding between customer sites the following tunnels need to be established:

- R1-R5 (customer A)
- R1-R4 (customer A)
- R4-R5 (customer A)
- R1-R5 (customer B)

Note that each tunnel setup involves creating VPLS interfaces on both endpoints of tunnel.

Negotiation of VPLS tunnels is done by LDP protocol - both endpoints of tunnel exchange labels they are going to use for tunnel. Data forwarding in tunnel then happens by imposing 2 labels on packets: tunnel label and transport label - label that ensures traffic delivery to the other endpoint of tunnel.

VPLS tunnels are configured in /interface vpls menu. vpls-id parameter identifies VPLS tunnel and must be unique for every tunnel between this and remote peer.

The necessary setup:

- on R1:

```
/interface vpls add name=A1toA2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=A1toA3 remote-peer=9.9.9.4 mac-address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=B1toB2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:b1 vpls-id=11 disabled=no
```

- on R4:

```
/interface vpls add name=A3toA1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:a3 vpls-id=10 disabled=no
/interface vpls add name=A3toA2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:a3 vpls-id=10 disabled=no
```

- on R5:

```
/interface vpls add name=A2toA1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:a2 vpls-id=10 disabled=no
/interface vpls add name=A2toA3 remote-peer=9.9.9.4 mac-address=00:00:00:00:00:a2 vpls-id=10 disabled=no
/interface vpls add name=B2toB1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:b2 vpls-id=11 disabled=no
```

Configuring VPLS tunnel causes dynamic LDP neighbor to be created and "targeted" LDP session to be established. Targeted LDP session is session that is established between two routers that are not direct neighbors. After this setup R1 LDP neighbors are:

```
[admin@R1] > mpls ldp neighbor pr
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0                 no            1.1.1.2
```

```
                                                                          2.2.2.2
                                                                          9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0                 yes    4.4.4.5
                                                                          5.5.5.5
                                                                          9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0                 yes    3.3.3.4
                                                                          5.5.5.4
                                                                          9.9.9.4
```

Note that labels for IP routes are also exchanged between VPLS peers, although there is no chance any of them will be used. For example, without adding additional links, R4 will not become next hop for any route on R1, so labels learned from R4 are not likely to be ever used. Still routers maintain all labels exchanged so that they are ready for use immediately if needed. This default behaviour can be overridden by filtering which is discussed later.

By monitoring state of VPLS interface its related information can be viewed:

```
[admin@R1] /interface vpls> monitor A1toA3 once
    remote-label: 24
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 1.1.1.2
  imposed-labels: 21,24
```

Here we can see that R1 has assigned label 27 for tunnel between R1 and R4. MPLS forwarding table shows that this label is recognized and instead of fowarding to some next hop, received over this tunnel:

```
[admin@R1] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE          NEXTHOP
 ...
11 27                                       A1toA3
 ...
```

In turn remote endpoint (R4) has assigned label 24.

igp-prefix shows route that is used to get to remote endpoint of tunnel. This implies that when forwarding traffic to remote endpoint of tunnel this router will impose transport label - label distributed by next hop (which is shown as igp-nexthop) to 9.9.9.4/32 for 9.9.9.4/32 route. This can be confirmed on R2:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE          NEXTHOP
 ...
 5 21                   18                   9.9.9.4/32         ether2             2.2.2.3
 ...
```

Tunnel label imposed on packets will be as assigned by remote router (R4) for this tunnel.

imposed-labels reflect this setup: packets produced by tunnel will have 2 labels on them: 21 and 24.

Penultimate hop popping effects on VPLS tunnels

Penultimate hop popping of transport label causes packets to arrive at VPLS tunnel endpoint with just one tag - tunnel tag. This makes VPLS tunnel endpoint do just one label lookup to find out what to do with packet. Transport label behaviour can be observed by traceroute tool between tunnel endpoints. For example, traceroute from R1 to R4 looks like this:

```
[admin@R1] > /tool traceroute 9.9.9.4 src-address=9.9.9.1
     ADDRESS                                    STATUS
   1         1.1.1.2 7ms 5ms 3ms
                      mpls-label=21
   2         2.2.2.3 5ms 4ms 18ms
                      mpls-label=18
   3         9.9.9.4 4ms 5ms 3ms
```

traceroute output shows, that endpoint of tunnel is receiving probe without label. The same happens with VPLS tunnel traffic - at R3 transport label (18) is popped and packet is switched with just tunnel label on.

The requirement to deliver packet with tunnel label to endpoint of tunnel explains configuration advice to use "loopback" IP addresses as tunnel endpoints. If in this case R4 was establishing LDP sessions with its address 3.3.3.4, penultimate hop popping would happen not at R3, but at R2, because R3 has network 3.3.3.0/24 as its connected network (and therefore advertises implicit null label). This would cause R3 (and not R4) to receive packet with just tunnel label on, yielding unpredicted results - either dropping frame if R3 does not recognize the packet or forwarding it the wrong way.

Another issue is having VPLS tunnel endpoints directly connected, as in case of R4 and R5. There are no transport labels they can use between themselves, because they both instruct other one to be penultimate hop popping router for their tunnel endpoint address. For example on R5:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
 3 AD 9.9.9.4/32         5.5.5.4         impl-null  9.9.9.4:0
 ...
```

This causes VPLS tunnel to use only tunnel label when sending packets:

```
[admin@R5] > /int vpls monitor A2toA3 once
    remote-label: 23
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 5.5.5.4
  imposed-labels: 23
```

Bridging ethernet segments with VPLS

VPLS tunnels provide virtual ethernet link between routers. To transparrently connect two physical ethernet segments, they must be bridged with VPLS tunnel. In general it gets done the same way as with EoIP interfaces.

So to transparently bridge customer B networks connected to R1 and R5 the following commands are used on R1:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether1
/interface bridge port add bridge=B interface=B1toB2
```

and on R5:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether3
/interface bridge port add bridge=B interface=B2toB1
```

Note that there is not need to run (R)STP protocol on bridge as there are links between segments B1 and B2 except single VPLS tunnel between R1 and R5.

Split horizon bridging

In the example setup there are 3 tunnels set up to connect segments A1, A2 and A3, establishing so called "full mesh" of tunnels between involved segments. If bridging without (R)STP was enabled, traffic loop would occur. There are a few solutions to this:

- enabling (R)STP to eliminate the loop. This approach has a drawback - (R)STP protocol would disable forwarding through one of tunnels and keep it just for backup purposes. That way traffic between 2 segments would have to go through 2 tunnels, making setup inefficent
- using bridge firewall to make sure that traffic does not get looped - involves firewall rule setup making bridging less efficent
- using bridge horizon feature

The basic idea of split horizon bridging is to make traffic arriving over some port never be sent out some set of ports. For VPLS purposes this would mean never sending packet that arrived over one VPLS tunnel over to other VPLS tunnel, as it is known in advance, that sender of packet has connection to target network itself.

For example, if device in A1 sent packet to broadcast or unknown MAC address (which causes bridges to flood all interfaces), it would get sent to both, R5 and R4 over VPLS tunnels. In regular setup e.g. R5 when receiving such packet over VPLS tunnel would send it in A2 connected to it and also over VPLS tunnel to R4. This way R4 would get 2 copies of the same packet and further cause traffic to loop.

Bridge horizon feature allows to configure bridge ports with horizon setting so that packet received over port with horizon value X is not forwarded or flooded to any port with the same horizon value X. So in case of full mesh of VPLS tunnels, each router must be configured with the same horizon value for VPLS tunnels that are bridged together.

For example, configuration commands for R1 to enable bridging for customer A are:

```
/interface bridge add name=A
/interface bridge port add bridge=A interface=A1toA2 horizon=1
/interface bridge port add bridge=A interface=A1toA3 horizon=1
```

In similar way bridge should be configured on R4 and R5. Note that physical ethernet port is not configured with horizon value. If it was, that would disabled bridge forwarding data at all.

Note that horizon value has meaning only locally - it does not get transmitted over network, therefore it does not matter if the same value is configured in all routers participating in bridged network.

Optimizing label distribution

Label binding filtering

During implementation of given example setup, it has become clear that not all label bindings are necessary. For example, there is no need to exchange IP route label bindings between R1 and R5 or R1 and R4, as there is no chance they will ever be used. Also, if given network core is providing connectivity only for mentioned customer ethernet segments, there is no real use to distribute labels for networks that connect routers between themselves, the only routes that matter are /32 routes to endpoints of VPLS tunnels.

Label binding filtering can be used to distribute only specified sets of labels to reduce resource usage and network load.

There are 2 types of label binding filters:

- which label bindings should be advertised to LDP neighbors, configured in /mpls ldp advertise-filter
- which label bindings should be accepted from LDP neighbors, configured in /mpls ldp accept-filter

Filters are organized in ordered list, specifying prefix that must include the prefix that is tested against filter and neighbor (or wildcard).

In given example setup all routers can be configured so that they advertise labels only for routes that allow to reach endpoints of tunnels. For this 2 advertise filters need to be configured on all routers:

```
/mpls ldp advertise-filter add prefix=9.9.9.0/24 advertise=yes
/mpls ldp advertise-filter add prefix=0.0.0.0/0 advertise=no
```

This filter causes routers to advertise only bindings for routes that are included by 9.9.9.0/24 prefix which covers tunnel endpoints (9.9.9.1/32, 9.9.9.4/32, 9.9.9.5/32). The second rule is necessary because default filter result, when no rule matches is to allow action in question.

In given setup there is no need to set up accept filter, because by convention introduced by 2 abovementioned rules no LDP router will distribute unnecessary bindings.

Note that filter changes do not affect existing mappings, so to take filter into effect, connections between neighbors need to be reset. This can get done by removing them:

```
[admin@R1] /mpls ldp neighbor> print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0                 no            1.1.1.2
                                                                                2.2.2.2
                                                                                9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0                 yes           4.4.4.5
                                                                                5.5.5.5
                                                                                9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0                 yes           3.3.3.4
                                                                                5.5.5.4
                                                                                9.9.9.4
[admin@R1] /mpls ldp neighbor> remove [find]
```

So on R1, for example we get:

```
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0  D 9.9.9.1/32                         30         9.9.9.5:0
 1  D 9.9.9.5/32                         impl-null  9.9.9.5:0
 2  D 9.9.9.4/32                         31         9.9.9.5:0
 3  D 9.9.9.2/32                         32         9.9.9.5:0
 4  D 9.9.9.3/32                         33         9.9.9.5:0
 5 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 6  D 9.9.9.1/32                         24         9.9.9.2:0
 7 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 8 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
 9 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
10  D 9.9.9.1/32                         27         9.9.9.4:0
11  D 9.9.9.5/32                         28         9.9.9.4:0
12  D 9.9.9.4/32                         impl-null  9.9.9.4:0
13  D 9.9.9.2/32                         29         9.9.9.4:0
14  D 9.9.9.3/32                         30         9.9.9.4:0
```

There still are unnecessary bindings, this time - the bindings distributed due to establishing targeted LDP session with remote endpoints of VPLS tunnels (bindings from 9.9.9.5 and 9.9.9.4)

To filter out those, we configure routers to not distribute any IP bindings to any of tunnel endpoint routers. For example on R1, filter table should look like this:

```
[admin@R1] /mpls ldp advertise-filter> print
Flags: X - disabled
 #   PREFIX             NEIGHBOR        ADVERTISE
 0   0.0.0.0/0          9.9.9.4         no
 1   0.0.0.0/0          9.9.9.5         no
 2   9.9.9.0/24         all             yes
 3   0.0.0.0/0          all             no
```

This causes routers to have minimal label binding tables, for example on R1:

```
[admin@R1] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 9.9.9.1/32         impl-null  9.9.9.2:0
 1 ADG  9.9.9.3/32         40         9.9.9.2:0
 2 ADG  9.9.9.4/32         41         9.9.9.2:0
 3 ADG  9.9.9.2/32         42         9.9.9.2:0
 4 ADG  9.9.9.5/32         43         9.9.9.2:0
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 1  D 9.9.9.1/32                         24         9.9.9.2:0
 2 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 3 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
```

```
 4 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
```

Note that IP binding distribution should not be disabled between R4 and R5 although they are tunnel endpoints. Doing so would not harm regular case, because R4 and R5 does not need IP bindings to VPLS tunnel data, but in case link between R3 and R5 would go down, all traffic to R5 from R1 would have to be rerouted through R4. In such case R5 not distributing IP bindings to R4 would cause R4 to not be able to forward MPLS traffic to R5.

Effects of label binding filtering on data forwarding in network

Note the traceroute results after these changes. Traceroute from R1 to R5 using R1 loopback address as source address still behaves the same - each hop reports received labels:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=9.9.9.1
     ADDRESS                                    STATUS
   1         1.1.1.2 11ms 4ms 5ms
                      mpls-label=27
   2         2.2.2.3 4ms 4ms 4ms
                      mpls-label=25
   3         9.9.9.5 12ms 3ms 3ms
```

But in case of traceroute using R1 address that faces network:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=1.1.1.1
     ADDRESS                                    STATUS
   1         0.0.0.0 timeout timeout timeout
   2         0.0.0.0 timeout timeout timeout
   3         9.9.9.5 3ms 3ms 3ms
```

Now all hops except the last one do not respond. The reason for this is the fact, that there is no label switching path back from R5 to R1 (which this time uses address 1.1.1.1) because there are no label bindings distributed - so ICMP response is routed and routers on the way back (R3 and R2) receive packets with their own source address and drop them right away without routing.

On the other hand, traceroute from R1 to R5 using their non-loopback addresses:

```
[admin@R1] > tool traceroute 4.4.4.5 src-address=1.1.1.1
     ADDRESS                                    STATUS
   1         1.1.1.2 13ms 1ms 1ms
   2         2.2.2.3 3ms 2ms 2ms
   3         4.4.4.5 3ms 3ms 23ms
```

There is no label switching involved doing this traceroute and it works just like in network without MPLS at all.

Manual:Cisco VPLS

Applies to RouterOS: v3, v4

Overview

Since version 3.20 RouterOS implements features that provide compatibility with Cisco VPLS features:

- Cisco style static VPLS pseudowires (RFC 4447 FEC type 0x80)* Cisco VPLS BGP-based auto-discovery (draft-ietf-l2vpn-signaling-08)

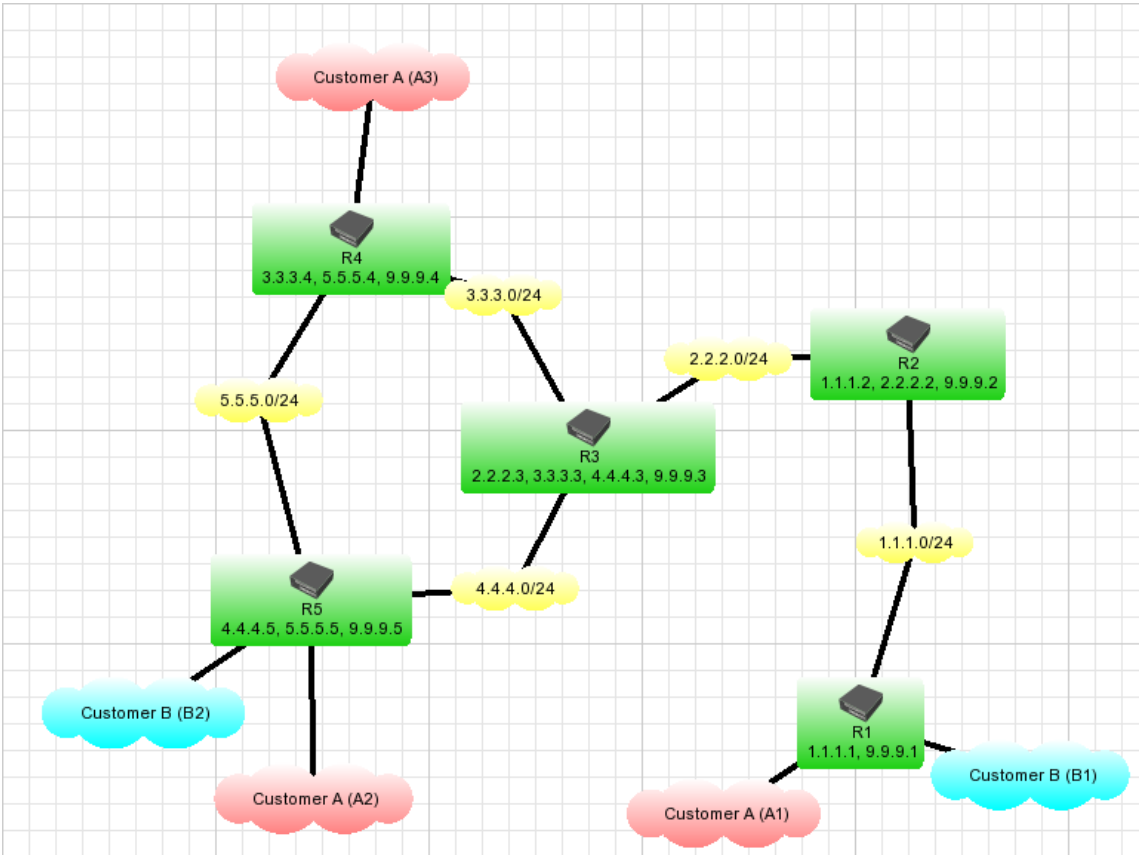
When signaling static VPLS tunnels (pseudowires) using LDP, Cisco does not use pseudowire endpoint identification as specified in RFC 4762 (FEC type 0x81), but uses other method - from RFC 4447 (FEC type 0x80). Such pseudowires can be configured in RouterOS by means of **cisco-style** and **cisco-style-id** settings.

Cisco does not implement BGP-based auto-discovery and signaling according to RFC 4671. Instead, Cisco implements BGP based auto-discovery (draft-ietf-l2vpn-signaling-08). This method specifies use of BGP only to auto-discover other peers that participate in VPLS. VPLS pseudowire signaling is done by LDP.

This document focuses on RouterOS configuration that is related to Cisco compatibility features, for general information on VPLS see MPLSVPLS, for information on RFC 4671 compatible BGP based VPLS, see BGP based VPLS.

Example network

The example network used throughout this document is the same as in MPLSVPLS.



The requirements of customers A and B are the same - ethernet segments must be transparently connected. Taking into account simplicity of given network topology Service Provider has decided to use R5 as route reflector and to have no backup route reflector. Consider that MPLS switching is configured and running, as discussed in MPLSVPLS, but no any VPLS configuration has been applied yet. the rest of this document deals with specifics that are introduced by using Cisco compatible VPLS features.

Customer's B networks are to be connected using static VPLS pseudowire, Customer's A networks are to be connected using VPLS BGP-based autodiscovery.

Configuring Cisco style static VPLS interface

Cisco compatible static VPLS interface is created by specifying appropriate settings: **cisco-style=yes** specifies that VPLS interface should use Cisco-like endpoint identification, parameter **cisco-style-id** specifies pseudowire ID to use. On R1:

```
[admin@R1] /interface vpls> add disabled=no cisco-style=yes cisco-style-id=666 remote-peer=9.9.9.5
```

and on R5:

```
[admin@R5] /interface vpls> add disabled=no cisco-style=yes cisco-style-id=666 remote-peer=9.9.9.1
```

This should result in establishment of targeted LDP session between R1 and R5 and VPLS interface becoming active:

```
[admin@R1] > mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0                 no            1.1.1.2
                                                                                2.2.2.2
                                                                                9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0                 yes           4.4.4.5
                                                                                5.5.5.5
                                                                                9.9.9.5
[admin@R1] > /interface vpls print
Flags: X - disabled, R - running, D - dynamic, B - bgp-signaled, C - cisco-bgp-signaled
 0 R   name="vpls1" mtu=1500 mac-address=02:94:02:DB:60:6E arp=enabled disable-running-check=no
remote-peer=9.9.9.5 cisco-style=yes cisco-style-id=666
[admin@R1] > /interface vpls monitor vpls1 once
       remote-label: 29
        local-label: 31
      remote-status:
          transport: 9.9.9.5/32
  transport-nexthop: 1.1.1.2
     imposed-labels: 24,29
```

The rest of configuration to enable transparent bridging of Customer B networks (configuring bridging) is the same as described in MPLSVPLS

Configuring BGP for Cisco compatible VPLS

Configuring Cisco compatible BGP VPLS instance makes router advertise VPLS NLRI according to draft-ietf-l2vpn-signaling-08. Note that this NLRI uses the same BGP AFI/SAFI values as RFC 4762 compatible BGP VPLS. Therefore only one of those specifications (configured as BGP peer address-families) can be supported at a time.

In order to avoid configuring full mesh of BGP sessions between routers acting as attachment points for customer A networks (R1,R4,R5), R5 will be used as route reflector - the same as in BGP based VPLS.

On R5, make sure that client-to-client-reflection is enabled:

```
[admin@R5] /routing bgp instance> pr
Flags: X - disabled
 0   name="default" as=65530 router-id=0.0.0.0 redistribute-connected=no redistribute-static=no redistribute-rip=no
     redistribute-ospf=no redistribute-other-bgp=no out-filter="" client-to-client-reflection=yes ignore-as-path-len=no
```

Create BGP peers with support for **l2vpn-cisco** in peers **address-families**, on R5 configure route reflection:

```
[admin@R5] /routing bgp peer> add remote-as=65530 update-source=lobridge instance=default remote-address=9.9.9.1
address-families=l2vpn-cisco route-reflect=yes
[admin@R5] /routing bgp peer> add remote-as=65530 update-source=lobridge instance=default remote-address=9.9.9.4
address-families=l2vpn-cisco route-reflect=yes
```

```
[admin@R1] /routing bgp peer> add remote-as=65530 update-source=lobridge instance=default remote-address=9.9.9.5
address-families=l2vpn-cisco
```

```
[admin@R4] /routing bgp peer> add remote-as=65530 update-source=lobridge instance=default remote-address=9.9.9.5
address-families=l2vpn-cisco
```

Configuring Cisco compatible BGP VPLS instance

In order for full mesh of VPLS pseudowires to get established, appropriate Cisco compatible VPLS instance must be created. Creating such instance makes router to inject VPLS BGP NLRI in BGP network:

```
[admin@R1] /interface vpls cisco-bgp-vpls> add bridge=A bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1
l2router-id=9.9.9.1 route-distinguisher=1:1 vpls-id=1:1
```

```
[admin@R4] /interface vpls cisco-bgp-vpls> add bridge=A bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1
l2router-id=9.9.9.4 route-distinguisher=1:1 vpls-id=1:1
```

```
[admin@R5] /interface vpls cisco-bgp-vpls> add bridge=A bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1
l2router-id=9.9.9.5 route-distinguisher=1:1 vpls-id=1:1
```

This causes full mesh of targeted LDP sessions to get established and appropriate VPLS interfaces created, e.g. on R4:

```
[admin@R4] > /mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 ...
 1 DOTV 9.9.9.5         9.9.9.4         9.9.9.5:0                 no            4.4.4.5
                                                                                  5.5.5.5
                                                                                  9.9.9.5
 2 DOTV 9.9.9.1         9.9.9.4         9.9.9.1:0                 yes           1.1.1.1
                                                                                  9.9.9.1
```

```
[admin@R4] > /interface vpls print
Flags: X - disabled, R - running, D - dynamic, B - bgp-signaled, C - cisco-bgp-signaled
 0 RDC name="vpls1" mtu=1500 mac-address=02:62:65:24:4C:FD arp=enabled disable-running-check=no remote-peer=9.9.9.1
       vpls-id=1:1 cisco-style=no cisco-style-id=0 vpls=cisco-bgp-vpls1

 1 RDC name="vpls2" mtu=1500 mac-address=02:58:9F:80:EB:94 arp=enabled disable-running-check=no remote-peer=9.9.9.5
       vpls-id=1:1 cisco-style=no cisco-style-id=0 vpls=cisco-bgp-vpls1
```

Manual:BGP based VPLS

Overview

MPLSVPLS page covers general introduction to VPLS service and configuration of LDP based VPLS tunnels. Due to their static nature LDP based VPLS tunnels have scalability issues that arise when number of VPLSes and sites participating in VPLSes grow. One of the problems is the requirement to maintan full mesh of LDP tunnels between sites forming VPLS. In case number of sites in VPLS is high, adding new site to existing VPLS can become burdensome for network administrator.

BGP based autodiscovery and signaling of VPLS tunnels can help to avoid complexity of configuration at the expense of running BGP protocol between VPLS routers. In general, BGP based VPLS serves two purposes:

- autodiscovery: there is no need to configure each VPLS router with all remote endpoints of VPLS tunnels, provided there are means to deliver BGP multiprotocol NLRIs between them - routers figure out remote endpoints of tunnels from received BGP Updates;
- signaling: labels used for VPLS tunnels by remote endpoints are distributed in the same BGP Updates, this means there is no need for targeted LDP sessions between tunnel endpoints as in case of LDP signaled VPLS.

For example, if LDP signaled VPLS is used, adding new site to existing VPLS would mean configuring router that connects new site to establish tunnels with the rest of sites and also configure all other routers to establish tunnels with router connecting this new site. BGP based VPLS, if configured properly eliminates need to adjust configuration on all routers forming VPLS.

The requirement to exchange BGP NLRIs between VPLS routers means that either full mesh of BGP sessions need to be established among routers forming VPLS or route reflector must be used. In case full mesh of BGP sessions are established between VPLS routers, the benefits of BGP based VPLS over LDP signaled VPLS are questionable - when new site is added to VPLS, BGP peer configuration still needs to be entered on every router forming given VPLS. When BGP route reflector is used, adding new site to VPLS becomes more simple - router connecting new site must only peer with route reflector and no additional configuration is required on other routers. Taking into account that route reflector can also be one of routers forming VPLS, there is no need for additional separate equipment. Of course, scalability and availability concerns still must be taken into account - multiple route reflectors can be used for backup purposes as well as for distributing information load.

The drawback of running BGP based VPLS is requirement to configure BGP which requires that network administrator has at least basic understanding of BGP, its multiprotocol capabilities and route reflectors. Therefore it is advised to implement LDP signaled VPLS if amount of sites and VPLS networks is small, topology is more static - that is, benefits of using BGP are not obvious.

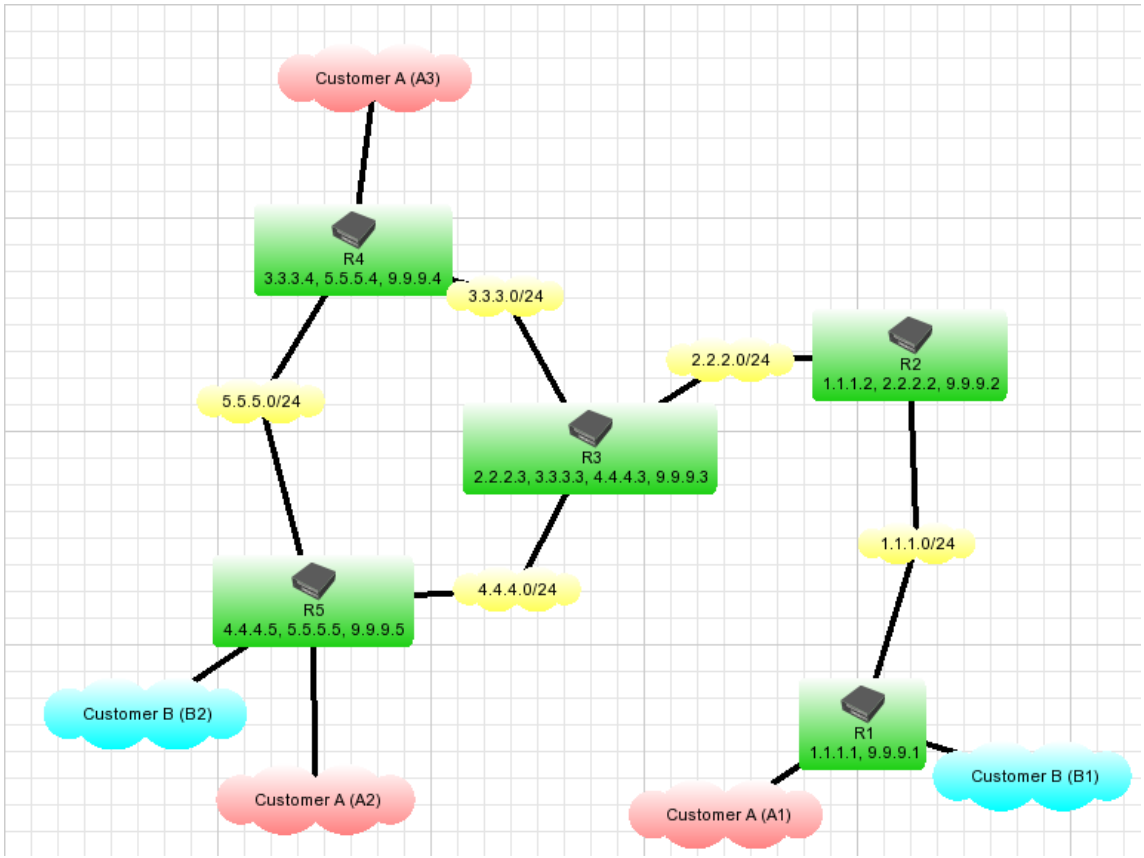
Note that BGP based VPLS is a method only for VPLS tunnel label exchange, it does not deal with delivery of traffic between VPLS tunnel endpoints, so general MPLS frame delivery between tunnel endpoints must be ensured as discussed in MPLSVPLS.

Suggested reading material:

- RFC 4761, Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling
- RFC 4456, BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)

Example network

Consider the same network as used for LDP signaled VPLS example in MPLSVPLS:



The requirements of customers A and B are the same - ethernet segments must be transparently connected. Taking into account simplicity of given network topology Service Provider has decided to use R5 as route reflector and to have no backup route reflector. Consider that MPLS switching is configured and running, as discussed in MPLSVPLS, but no any VPLS configuration has been applied yet. the rest of this document deals with specifics that are introduced by use of BGP for VPLS signaling.

Configuring IBGP session for VPLS signaling

At first, BGP instance must be configured, default instance can also be used:

```
[admin@R1] /routing bgp instance> print
Flags: X - disabled
 0   name="default" as=65530 router-id=0.0.0.0 redistribute-connected=no redistribute-static=no
     redistribute-rip=no redistribute-ospf=no redistribute-other-bgp=no out-filter=""
     client-to-client-reflection=yes ignore-as-path-len=no
```

To enable VPLS NLRI delivery across BGP, BGP multiprotocol capability must be used. This is enabled by specifying **l2vpn** in BGP peer's **address-families** setting.

For example, to configure BGP connection between R1 and R5, the following commands should get issued.

On R1:

```
[admin@R1] /routing bgp peer> add remote-address=9.9.9.5 remote-as=65530 address-families=l2vpn \
  update-source=lobridge
```

and on R5:

```
[admin@R5] /routing bgp peer> add remote-address=9.9.9.1 remote-as=65530 address-families=l2vpn \
  update-source=lobridge
```

BGP connection should get established between R1 and R5. This can be confirmed by:

```
[admin@R1] /routing bgp peer> print status
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.5 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=4.4.4.5
     local-address=9.9.9.1 uptime=3s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established
```

There are several things to note about BGP peer configuration:

- there is no need to distribute any IP or IPv6 routes and even no need have IP or IP6 support over BGP connection at all to be able to exchange VPLS NLRIs, it is sufficient to specify **address-families=l2vpn**
- "loopback" addresses of routers are used as BGP peer addresses (local address is configured by means of **update-source** setting). BGP peer, when originating VPLS NLRI, specifies its local address as BGP NextHop (for example, in given setup R1 originating BGP NLRIs will use address 9.9.9.1 as BGP NextHop address), receiving VPLS router uses received BGP NextHop address as tunnel endpoint address and therefore uses transport label that ensures delivery to BGP NextHop. In order for penultimate hop popping to work properly, it is advised to use loopback IP address for this. See penultimate hop popping related discussion in MPLSVPLS.

Configuring Route Reflector

In its simplest sense BGP Route Reflector re-advertises received IBGP routes without changing BGP NextHop for route. This feature can be used to avoid setting up full mesh of BGP connections. Note that for router be able to operate as route reflector for VPLS NLRIs, it is not necessary for it to participate in any VPLS, it is even not necessary for it to have MPLS support. Still it is mandatory for VPLS routers to be able to establish BGP sessions with route reflector, therefore IP connectivity is a must.

Route reflector's BGP instance must be configured with **client-to-client-reflection=yes** setting:

```
[admin@R5] /routing bgp instance> print
Flags: X - disabled
 0   name="default" as=65530 router-id=0.0.0.0 redistribute-connected=no redistribute-static=no
     redistribute-rip=no redistribute-ospf=no redistribute-other-bgp=no out-filter=""
     client-to-client-reflection=yes ignore-as-path-len=no
```

Additionaly, peers on route reflector must be configured with **route-reflect=yes** setting:

```
[admin@R5] /routing bgp peer> print
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.1 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge
```

```
[admin@R5] /routing bgp peer> set 0 route-reflect=yes
[admin@R5] /routing bgp peer> print
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.1 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=yes hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge
```

To enable R5 to operate as route reflector, all its peers should get added with **route-reflect=yes** setting. So to enable proper VPLS NLRI distribution, R5 must be configured with 2 BGP peers - R1 and R4:

```
[admin@R5] /routing bgp peer> print status
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.1 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=yes hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=1.1.1.1
     local-address=9.9.9.5 uptime=5m55s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established

 1   name="peer2" instance=default remote-address=9.9.9.4 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=yes hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=3.3.3.4
     local-address=9.9.9.5 uptime=23s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established
```

But R1 and R4 must only peer with R5. On R1:

```
[admin@R1] /routing bgp peer> print status
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.5 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=4.4.4.5
     local-address=9.9.9.1 uptime=6m33s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established
```

and on R4:

```
[admin@R4] /routing bgp peer> print status
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.5 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=4.4.4.5
     local-address=9.9.9.4 uptime=3s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established
```

Using route reflector means that in order to add new site to some VPLS, e.g. connected by router Ry, would mean adding Ry as BGP peer to R5 (with route-reflect=yes setting) and adding R5 as BGP peer to Ry.

Configuring BGP signaled VPLS

Configuring ethernet bridging

BGP signalled VPLS tunnels are created dynamically when proper BGP NLRIs are received. Therefore there is no need to configure any VPLS interfaces. Still, to transparently deliver packets from ethernet segment across VPLS bridging must be configured. For example, on R1 two bridges are created, named "A" and "B" with appropriate customer-facing ethernet interfaces added to them:

```
[admin@R1] /interface bridge> print
Flags: X - disabled, R - running
 0  R name="lobridge" mtu=1500 arp=enabled mac-address=00:00:00:00:00:00 protocol-mode=none
      priority=0x8000 auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s
      forward-delay=15s transmit-hold-count=6 ageing-time=5m

 1  R name="A" mtu=1500 arp=enabled mac-address=00:01:50:E7:00:09 protocol-mode=none
      auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s forward-delay=15s
      priority=0x8000 transmit-hold-count=6 ageing-time=5m

 2  R name="B" mtu=1500 arp=enabled mac-address=00:01:50:E7:00:08 protocol-mode=none
      auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s forward-delay=15s
      priority=0x8000 transmit-hold-count=6 ageing-time=5m
[admin@R1] /interface bridge> port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE                           BRIDGE PRIORITY PATH-COST            HORIZON
 0    ether2                              A      0x80     10                   none
 1    ether1                              B      0x80     10                   none
```

Configuring BGP signaled VPLS instances

Configuring BGP signaled VPLS instance makes router advertise VPLS BGP NLRI that advertises that particular router belongs to some VPLS. Upon receiving such advertisement, other members of same VPLS know to establish VPLS tunnel with this router.

To configure VPLS for customers A and B, on R1 the following commands should be issued:

```
[admin@R1] /interface vpls bgp-vpls> add bridge=A bridge-horizon=1 route-distinguisher=1:1 \
   site-id=1 import-route-targets=1:1 export-route-targets=1:1
[admin@R1] /interface vpls bgp-vpls> add bridge=B bridge-horizon=1 route-distinguisher=2:2 \
   site-id=1 import-route-targets=2:2 export-route-targets=2:2
```

Note: Since v3.20 **vpls-id** was replaced with separate import/export-route-targets to provide more flexibility.

route-distinguisher setting specifies value that gets attached to VPLS NLRI so that receiving routers can distinguish advertisements that may otherwise look the same. This implies that unique route-distinguisher for every VPLS must be used. It is not necessary to use the same route distinguisher for some VPLS on all routers forming that VPLS as distinguisher is not used for determining if some BGP NLRI is related to particular VPLS (Route Target attribute is used for this), but it is mandatory to have different distinguishers for different VPLSes.

export-route-targets setting is used for tagging BGP NLRI

import-route-targets setting is used to determine if BGP NLRI is related to particular VPLS

site-id setting must be unique among members of particular VPLS. It is advisable although not mandatory to allocate site-id values in as narrow range as possible as that increases efficency of BGP (for details see RFC 4761).

bridge setting specifies bridge to which dynamically created VPLS tunnels should get added.

bridge-horizon specifies horizon value to be used for ports added to bridge (see Split horizon bridging discussion in MPLSVPLS).

After configuring R4 as member of VPLS 1:1 (used for customer A) with command:

```
[admin@R4] /interface vpls bgp-vpls> add bridge=A bridge-horizon=1 route-distinguisher=1:1 \
   site-id=4 import-route-targets=1:1 export-route-targets=1:1
```

Dynamic VPLS tunnel gets created on both R1 and R4. On R1 this can be confirmed:

```
[admin@R1] > /interface vpls print
Flags: X - disabled, D - dynamic, R - running, B - bgp-signaled
 0 RDB name="vpls1" mtu=1500 mac-address=02:FA:33:C4:7A:A9 arp=enabled
       disable-running-check=no remote-peer=9.9.9.4 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
[admin@R1] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE                           BRIDGE PRIORITY PATH-COST            HORIZON
 0    ether2                              A      0x80     10                   none
 1    ether1                              B      0x80     10                   none
 2  D vpls1                               A      0x80     50                   1
```

Here we have confirmed also that route reflection as configured on R5 works as expected as there is no BGP peer relationship between R1 and R4.

Additionally we must configure R5 to participate in VPLS for customer A:

```
[admin@R5] /interface vpls bgp-vpls> add bridge=A bridge-horizon=1 route-distinguisher=1:1 \
   site-id=5 import-route-targets=1:1 export-route-targets=1:1
```

This causes R1 and R4 to establish additional VPLS tunnel with R5. For example on R1:

```
[admin@R1] > /interface vpls print
Flags: X - disabled, D - dynamic, R - running, B - bgp-signaled
 0 RDB name="vpls1" mtu=1500 mac-address=02:FA:33:C4:7A:A9 arp=enabled
       disable-running-check=no remote-peer=9.9.9.4 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
 1 RDB name="vpls2" mtu=1500 mac-address=02:FF:B7:0E:4B:97 arp=enabled
       disable-running-check=no remote-peer=9.9.9.5 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
```

And bridge port to get added with proper horizon value:

```
[admin@R1] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE                           BRIDGE PRIORITY PATH-COST            HORIZON
 0    ether2                              A      0x80     10                   none
 1    ether1                              B      0x80     10                   none
 2  D vpls1                               A      0x80     50                   1
 3  D vpls2                               A      0x80     50                   1
```

To complete the setup, necessary configuration for customer B VPLS should be applied to R5:

```
[admin@R5] /interface vpls bgp-vpls> add site-id=5 route-distinguisher=2:2 bridge=B \
    bridge-horizon=1 import-route-targets=2:2 export-route-targets=2:2
```

As the result we get full mesh of VPLS tunnels established, for example on R5:

```
[admin@R5] /interface vpls> print
Flags: X - disabled, D - dynamic, R - running, B - bgp-signaled

 0 RDB name="vpls1" mtu=1500 mac-address=02:FA:5C:28:29:D3 arp=enabled
       disable-running-check=no remote-peer=9.9.9.1 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
 1 RDB name="vpls2" mtu=1500 mac-address=02:EA:51:31:3E:2B arp=enabled
       disable-running-check=no remote-peer=9.9.9.4 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
 2 RDB name="vpls3" mtu=1500 mac-address=02:F6:CF:06:1E:CB arp=enabled
       disable-running-check=no remote-peer=9.9.9.1 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls2
```

Note that remote-peer for VPLS tunnels is BGP NextHop address as received in BGP Update. For example BGP logs on R5 when receiving Update for VPLS 2:2 (customer B), say:

```
11:24:06 route,bgp,debug,packet UPDATE Message
11:24:06 route,bgp,debug,packet     RemoteAddress=9.9.9.1
11:24:06 route,bgp,debug,packet     MessageLength=79
11:24:06 route,bgp,debug,packet
11:24:06 route,bgp,debug,packet     PathAttributes
11:24:06 route,bgp,debug,packet         bgp-origin=INCOMPLETE
11:24:06 route,bgp,debug,packet         bgp-nexthop=9.9.9.1
11:24:06 route,bgp,debug,packet         bgp-localpref=100
11:24:06 route,bgp,debug,packet         bgp-extended-communities=RT:2:2
11:24:06 route,bgp,debug,packet
11:24:06 route,bgp,debug,packet     NLRI= rd
11:24:06 route,bgp,debug,packet         type=0
11:24:06 route,bgp,debug,packet         administrator=2
11:24:06 route,bgp,debug,packet         assigned-number=2 veId=1 veBlockOffset=0 veBlockSize=16
                            labelBase=40
```

This is reflected for dynamic VPLS tunnel, where remote-peer for tunnel with export-route-targer 2:2 is 9.9.9.1. This implies that R5 uses IGP route that leads to 9.9.9.1 to decide what transport label to use. In given case there are /32 IGP routes distributed in the network by means of OSPF, therefore:

```
[admin@R5] /interface vpls> monitor 2 once
    remote-label: 45
     local-label: 40
   remote-status:
      igp-prefix: 9.9.9.1/32
     igp-nexthop: 4.4.4.3
  imposed-labels: 17,45
```

Shows that 9.9.9.1/32 route is used and immediate nexthop is 4.4.4.3. Labels attached to VPLS packets are 17 and 45 where 45 is label mapping received with BGP Update, and 17 is label assigned by R3 for prefix 9.9.9.1/32:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS         NEXTHOP          LABEL      PEER
 ...
14 AD 9.9.9.1/32          4.4.4.3          17         9.9.9.3:0
 ...
```

Manual:VPLS Control Word

Summary

Standards: `Pseudowire Fragmentation and Reassembly (RFC 4623)`

VPLS allows remote sites to share an Ethernet broadcast domain by connecting sites through **pseudo-wires(PW)** tunnels over a packet switching network (PSN). Since VPLS encapsulation adds additional overhead, each interface in LSP should be able to transmit large enough packet.

Each ethernet chipset has hardware limitation on maximum packet size that it can transmit. Even now there are Ethernets that supports only one Vlan tag, meaning that maximum packet size without Ethernet header and checksum (L2MTU) is 1504 bytes. Obviously it is not enough to forward VPLS encapsulated Ethernet frame without fragmentation (at least 1524 L2MTU support is required). See MTU article for maximum supported L2MTUs on RouterBOADs.

Since not even all RouterBOARDs support enough L2MTU to transmit VPLS encapsulated packet without fragmentation, RouterOS have added Pseudowire Fragmentation and Reassembly (PWE3) support according to RFC 4623 using 4-byte **Control Word (CW)**.

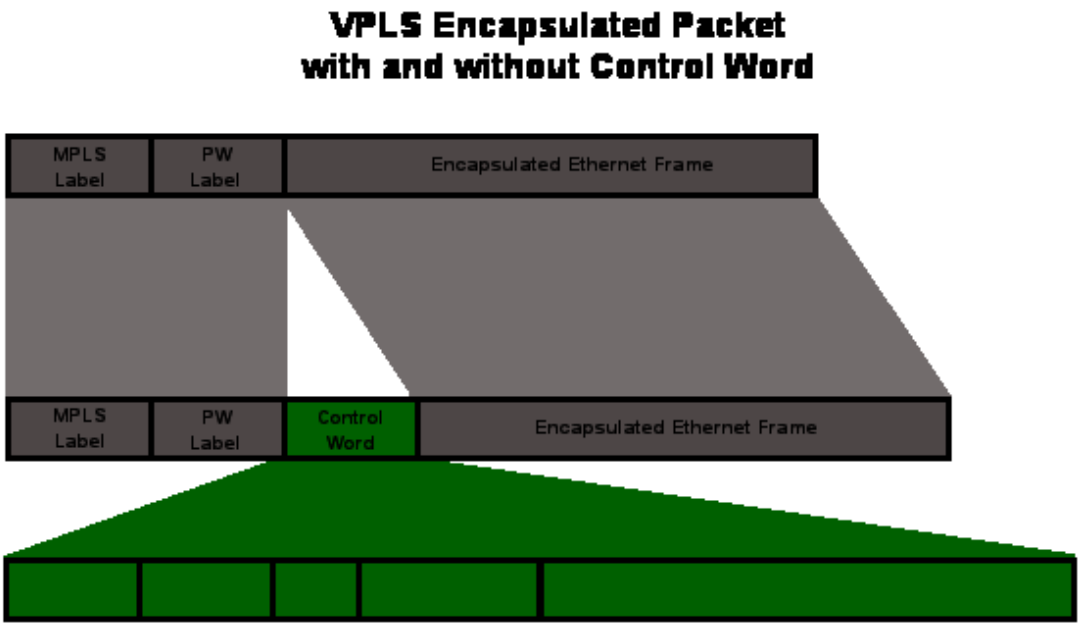
Control Word Usage

In RouterOS Control Word is used for packet fragmentation and reassembly inside VPLS tunnel and is done by utilizing optional **Control Word (CW)**. CW is added between PW label (demultiplexor) and packet payload and adds additional 4-byte overhead.

Note: Reordering OOO packets are not implemented, out of order fragments will be dropped

Until RouterOS v5.5 CW was used always, but, for compatibility with other vendors that do not use CW, feature to turn off Control Word usage was added. CW usage is controlled by one new parameter `use-control-word` in /interface vpls bgp-vpls and /interface vpls cisco-bgp-vpls

VPLS Packet and CW Format



As you can see **Control Word** is divided into 5 fileds:

- 0000 - 4-bits identifies that packet packet is PW (not IP)
- Flags - 4bits
- Frag - 2bits value that indicates payload fragmentation.
- Len - 6bits
- Seq - 16bits sequence number used to detect packet loss / misordering.

According to RFC generation and processing of sequence number is optional.

Example Setup

To show CW usage we will use simple three router setup as illustrated below.

This setup will not explain BGP and LDP configuration since its detailed explanation is found in other articles. `Read here>>`

See Also

- Basic MPLS and LDP based VPLS
- BGP based VPLS
- VPLS with Cisco routers

[Top | Back to Content]

Manual:Virtual Routing and Forwarding

Applies to RouterOS: 3, v4

Packages required: ***routing-test***, ***mpls-test*** *for RouterOS v3;* ***routing***, ***mpls*** *for RouterOS v4+*

Description

RouterOS 3.x allows to create multiple Virtual Routing and Forwarding instances on a single router. This is useful for BGP based MPLS VPNs. Unlike BGP VPLS, which is OSI Layer 2 technology, BGP VRF VPNs work in Layer 3 and as such exchange IP prefixes between routers. VRFs solve the problem of overlapping IP prefixes, and provide the required privacy (via separated routing for different VPNs).

To create a VRF, configure it under **/ip route vrf**. You can now add routes to that VRF - simply specify **routing-mark** attribute. Connected routes from interfaces belonging to a VRF will be installed in the right routing table automatically.

Technically VRFs are based on policy routing. There is exactly one policy route table for each active VRF. The existing policy routing support in MT RouterOS is not changed; but on the other hand, it is not possible to have policy routing *within* a VRF. The main differences between VRF tables and simple policy routing are:

- Routes in VRF tables resolve next-hops in their own route table by default, while policy routes always use the main route table. Read-only route attribute **gateway-table** displays information about which table is used for a particular route (default is **main**).
- Route lookup is different. For policy routing: after route lookup has been done in policy-route table, and no route was found, route lookup proceeds to the main route table. For VRFs: if lookup is done, and no route is found in VRF route table, the lookup fails with "network unreachable" error. (You can still override this behavior with custom route lookup rules, as they have precedence.)

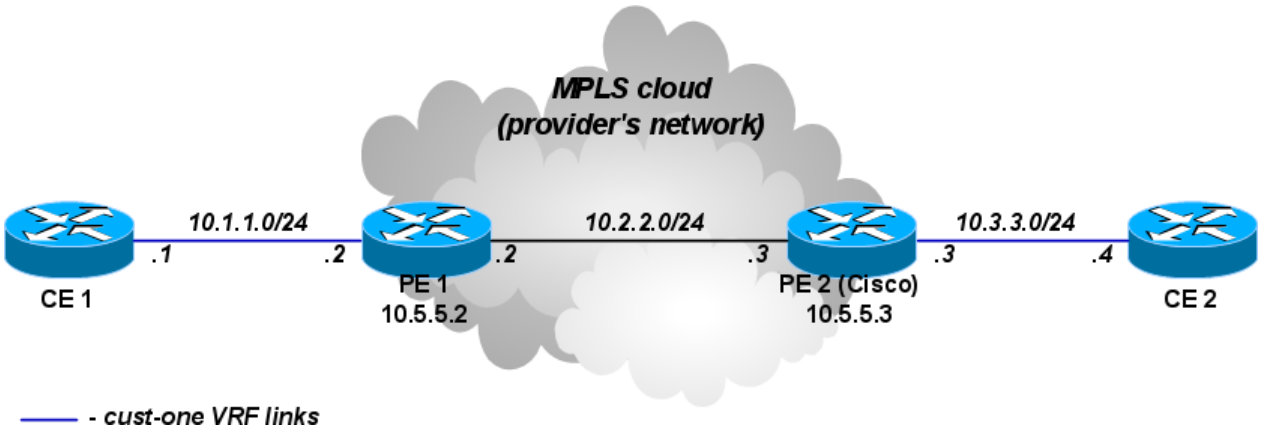
You can use multi-protocol BGP with VPNv4 address family to distribute routes from VRF route tables - not only to other routers, but also to different routing tables in the router itself. First configure the route distinguisher for a VRF. It can be done under **/ip route vrf**. Usually there will be one-to-one correspondence between route distinguishers and VRFs, but that's not a mandatory requirement. Route installation in VRF tables is controlled by BGP extended communities attribute. Configure import and export lists under **/ip route vrf**, **import-route-targets** and **export-route-targets**. Export route target list for a VRF should contained at least the route distinguisher for that VRF. Then configure a list of VRFs for each BGP instance that will participate in VRF routing.

Once list of VRFs for BGP instance, route distinguisher and export route targets has been configured, some active VPNv4 address family routes may be created, depending on BGP redistribution settings. They are installed in a separate route table and, if present, visible under **/routing bgp vpnv4-route**. These so called VPNv4 routes have prefix that consists of a route distinguisher and an IPv4 network prefix. This way you can have overlapping IPv4 prefixes distributed in BGP.

Please note that a VPNv4 route will be distributed only if it has a valid MPLS label. You need to install **mpls-test** package and configure valid label range for this to work. (Default configuration has valid label range.)

Examples

The simplest MPLS VPN setup



In this example rudimentary MPLS backbone (consisting of two Provider Edge (PE) routers PE1 and PE2) is created and configured to forward traffic between Customer Edge (CE) routers CE1 and CE2 routers that belong to *cust-one* VPN.

CE1 Router

```
/ip address add address=10.1.1.1/24 interface=ether1
# use static routing
/ip route add dst-address=10.3.3.0/24 gateway=10.1.1.2
```

CE2 Router

```
/ip address add address=10.3.3.4/24 interface=ether1
/ip route add dst-address=10.1.1.0/24 gateway=10.3.3.3
```

PE1 Router

```
/interface bridge add name=lobridge
/ip address add address=10.1.1.2/24 interface=ether1
/ip address add address=10.2.2.2/24 interface=ether2
/ip address add address=10.5.5.2/32 interface=lobridge
/ip route vrf add disabled=no routing-mark=cust-one route-distinguisher=1.1.1.1:111 \
    export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111 interfaces=ether1
/mpls ldp set enabled=yes transport-address=10.5.5.2
/mpls ldp interface add interface=ether2
/routing bgp instance set default as=65000
/routing bgp instance vrf add instance=default routing-mark=cust-one redistribute-connected=yes
/routing bgp peer add remote-address=10.5.5.3 remote-as=65000 address-families=vpnv4 \
    update-source=lobridge
 # add route to the remote BGP peer's loopback address
/ip route add dst-address=10.5.5.3/32 gateway=10.2.2.3
```

PE2 Router (Cisco)

```
ip vrf cust-one
 rd 1.1.1.1:111
 route-target export 1.1.1.1:111
 route-target import 1.1.1.1:111
 exit

interface Loopback0
 ip address 10.5.5.3 255.255.255.255

mpls ldp router-id Loopback0 force
mpls label protocol ldp

interface FastEthernet0/0
 ip address 10.2.2.3 255.255.255.0
 mpls ip

interface FastEthernet1/0
 ip vrf forwarding cust-one
 ip address 10.3.3.3 255.255.255.0

router bgp 65000
 neighbor 10.5.5.2 remote-as 65000
 neighbor 10.5.5.2 update-source Loopback0
 address-family vpnv4
  neighbor 10.5.5.2 activate
  neighbor 10.5.5.2 send-community both
  exit-address-family
 address-family ipv4 vrf cust-one
  redistribute connected
  exit-address-family

ip route 10.5.5.2 255.255.255.255 10.2.2.2
```

Results

Check that VPNv4 route redistribution is working:

```
[admin@PE1] > /routing bgp vpnv4-route print detail
Flags: L - label present
 0 L route-distinguisher=1.1.1.1:111 dst-address=10.3.3.0/24 gateway=10.5.5.3
     interface=ether2 in-label=17 out-label=17 bgp-local-pref=100 bgp-med=0
     bgp-origin=incomplete bgp-ext-communities="RT:1.1.1.1:111"

 1 L route-distinguisher=1.1.1.1:111 dst-address=10.1.1.0/24 interface=ether1
     in-label=16 bgp-ext-communities="RT:1.1.1.1:111"
```

Check that the 10.3.3.0 is installed in IP routes, in cust-one route table:

```
[admin@PE1] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADC  10.1.1.0/24         10.1.1.2         ether1              0
 1 ADb  10.3.3.0/24                         10.5.5.3 recursi... 20
 2 ADC  10.2.2.0/24         10.2.2.2         ether2              0
 3 ADC  10.5.5.2/32         10.5.5.2         lobridge            0
 4 A S  10.5.5.3/32                         10.2.2.3 reachab... 1
```

Let's take closer look at IP routes in cust-one VRF. The 10.1.1.0/24 IP prefix is a connected route that belongs to an interface that was configured to belong to cust-one VRF. The 10.3.3.0/24 IP prefix was advertised via BGP as VPNv4 route from PE2 and is imported in this VRF routing table, because our configured **import-route-targets** matched the BGP extended communities attribute it was advertised with.

```
[admin@PE1] /ip route> print detail where routing-mark=cust-one
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 ADC  dst-address=10.1.1.0/24 pref-src=10.1.1.2 gateway=ether1 distance=0 scope=10
        routing-mark=cust-one

 1 ADb  dst-address=10.3.3.0/24 gateway=10.5.5.3 recursive via 10.2.2.3 ether2
        distance=20 scope=40 target-scope=30 routing-mark=cust-one
        bgp-local-pref=100 bgp-origin=incomplete
        bgp-ext-communities="RT:1.1.1.1:111"
```

The same for Cisco:

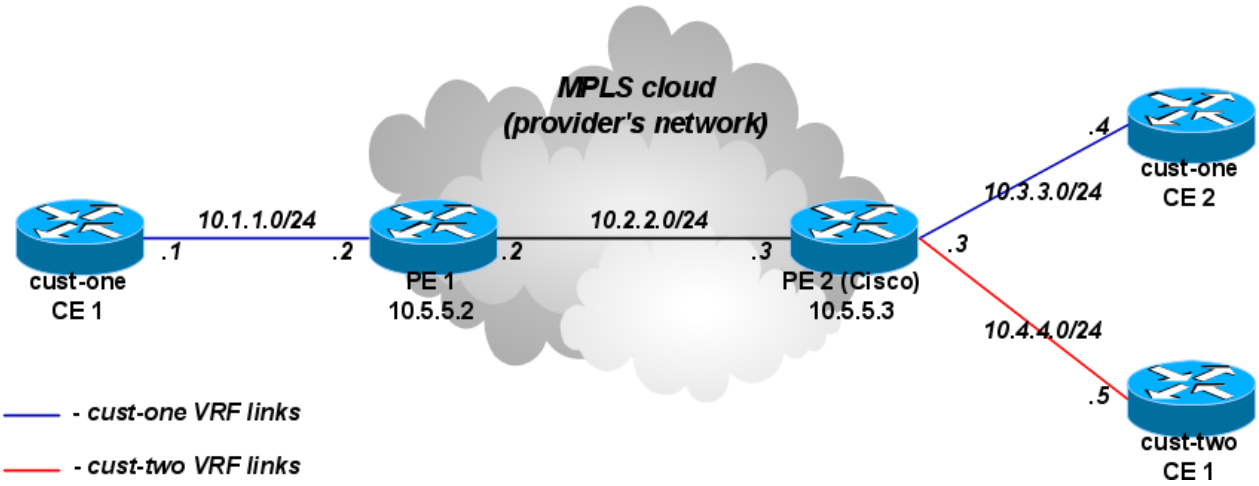
```
PE2#show ip bgp vpnv4 all
BGP table version is 5, local router ID is 10.5.5.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1.1.1.1:111 (default for vrf cust-one)
*>i10.1.1.0/24      10.5.5.2                      100      0 ?
*> 10.3.3.0/24      0.0.0.0                  0         32768 ?

PE2#show ip route vrf cust-one

Routing Table: cust-one
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
```

```
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
B       10.1.1.0 [200/0] via 10.5.5.2, 00:05:33
     10.0.0.0/24 is subnetted, 1 subnets
C       10.3.3.0 is directly connected, FastEthernet1/0
```

You should be able to ping from CE1 to CE2 and vice versa.

```
[admin@CE1] > /ping 10.3.3.4
10.3.3.4 64 byte ping: ttl=62 time=18 ms
10.3.3.4 64 byte ping: ttl=62 time=13 ms
10.3.3.4 64 byte ping: ttl=62 time=13 ms
10.3.3.4 64 byte ping: ttl=62 time=14 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 13/14.5/18 ms
```

A more complicated setup (changes only)



As opposed to the simplest setup, in this example we have two customers: cust-one and cust-two.

We configure two VPNs for then, cust-one and cust-two respectively, and exchange all routes between them. (This is also called "route leaking").

Note that this could be not the most typical setup, because routes are usually not exchanged between different customers. In contrast, by default it should not be possible to gain access from one VRF site to a different VRF site in another VPN. (This is the "Private" aspect of VPNs.) Separate routing is a way to provide privacy; and it is also required to solve the problem of overlapping IP network prefixes. Route exchange is in direct conflict with these two requirement but may sometimes be needed (e.g. temp. solution when two customers are migrating to single network infrastructure).

CE1 Router, *cust-one*

```
/ip route add dst-address=10.4.4.0/24 gateway=10.1.1.2
```

CE2 Router, *cust-one*

```
/ip route add dst-address=10.4.4.0/24 gateway=10.3.3.3
```

CE1 Router, *cust-two*

```
/ip address add address=10.4.4.5 interface=ether1
/ip route add dst-address=10.1.1.0/24 gateway=10.3.3.3
/ip route add dst-address=10.3.3.0/24 gateway=10.3.3.3
```

PE1 Router

```
# replace the old VRF with this:
/ip route vrf add disabled=no routing-mark=cust-one route-distinguisher=1.1.1.1:111 \
    export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111,2.2.2.2:222 interfaces=ether1
```

PE2 Router (Cisco)

```
ip vrf cust-one
 rd 1.1.1.1:111
 route-target export 1.1.1.1:111
 route-target import 1.1.1.1:111
 route-target import 2.2.2.2:222
 exit

ip vrf cust-two
 rd 2.2.2.2:222
 route-target export 2.2.2.2:222
 route-target import 1.1.1.1:111
 route-target import 2.2.2.2:222
 exit

interface FastEthernet2/0
 ip vrf forwarding cust-two
 ip address 10.4.4.3 255.255.255.0

router bgp 65000
 address-family ipv4 vrf cust-two
  redistribute connected
  exit-address-family
```

Variation: replace the Cisco with another MT

PE2 Mikrotik config

```
/interface bridge add name=lobridge
/ip address
 add address=10.2.2.3/24 interface=ether1
 add address=10.3.3.3/24 interface=ether2
 add address=10.4.4.3/24 interface=ether3
 add address=10.5.5.3/32 interface=lobridge
/ip route vrf
 add disabled=no routing-mark=cust-one route-distinguisher=1.1.1.1:111 \
     export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111,2.2.2.2:222 \
     interfaces=ether2
 add disabled=no routing-mark=cust-two route-distinguisher=2.2.2.2:222 \
    export-route-targets=2.2.2.2:222 import-route-targets=1.1.1.1:111,2.2.2.2:222 \
    interfaces=ether3
/mpls ldp set enabled=yes transport-address=10.5.5.3
/mpls ldp interface add interface=ether1
/routing bgp instance set default as=65000
/routing bgp instance vrf add instance=default routing-mark=cust-one redistribute-connected=yes
/routing bgp instance vrf add instance=default routing-mark=cust-two redistribute-connected=yes
/routing bgp peer add remote-address=10.5.5.2 remote-as=65000 address-families=vpnv4 \
    update-source=lobridge
 # add route to the remote BGP peer's loopback address
/ip route add dst-address=10.5.5.2/32 gateway=10.2.2.2
```

Results

The output of **/ip route print** now is interesting enough to deserve detailed observation.

```
[admin@PE2] /ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADb  10.1.1.0/24                        10.5.5.2 recurs... 20
 1 ADC  10.3.3.0/24        10.3.3.3        ether2             0
 2 ADb  10.4.4.0/24                                           20
 3 ADb  10.1.1.0/24                        10.5.5.2 recurs... 20
 4 ADb  10.3.3.0/24                                           20
 5 ADC  10.4.4.0/24        10.4.4.3        ether3             0
 6 ADC  10.2.2.0/24        10.2.2.3        ether1             0
 7 A S  10.5.5.2/32                        10.2.2.2 reacha... 1
 8 ADC  10.5.5.3/32        10.5.5.3        lobridge           0
```

The route 10.1.1.0/24 was received from remote BGP peer and is installed in both VRF routing tables.

The routes 10.3.3.0/24 and 10.4.4.0/24 are also installed in both VRF routing tables. Each is as connected route in one table and as BGP route in another table. This has nothing to do with their being advertised via BGP. They are

simply being "advertised" to local VPNv4 route table and locally reimported after that. Import and export **route-targets** determine in which tables they will end up.

This can be deduced from its attributes - they don't have the usual BGP properties. (Route 10.4.4.0/24.)

```
[admin@PE2] /ip route> print detail where routing-mark=cust-one
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 ADb  dst-address=10.1.1.0/24 gateway=10.5.5.2 recursive via 10.2.2.2 ether1
        distance=20 scope=40 target-scope=30 routing-mark=cust-one
        bgp-local-pref=100 bgp-origin=incomplete
        bgp-ext-communities="RT:1.1.1.1:111"

 1 ADC  dst-address=10.3.3.0/24 pref-src=10.3.3.3 gateway=ether2 distance=0 scope=10
        routing-mark=cust-one

 2 ADb  dst-address=10.4.4.0/24 distance=20 scope=40 target-scope=10
        routing-mark=cust-one bgp-ext-communities="RT:2.2.2.2:222"
```

Static inter-VRF routes

In general it is recommended that all routes between VRF should be exchanged using BGP local import and export functionality. If that is not enough, static routes can be used to achieve this so-called route leaking.

There are two ways to install a route that has gateway in different routing table than the route itself.

The first way is to explicitly specify routing table in gateway field when adding route. This is only possible for the "main" routing table. Example:

```
# add route to 5.5.5.0/24 in 'vrf1' routing table with gateway in the main routing table
add dst-address=5.5.5.0/24 gateway=10.3.0.1@main routing-mark=vrf1
```

The second way is to explicitly specify interface in gateway field. The interface specified can belong to a VRF instance. Example:

```
# add route to 5.5.5.0/24 in the main routing table with gateway at 'ether2' VRF interface
add dst-address=5.5.5.0/24 gateway=10.3.0.1%ether2 routing-mark=main

# add route to 5.5.5.0/24 in the main routing table with 'ptp-link-1' VRF interface as gateway
add dst-address=5.5.5.0/24 gateway=ptp-link-1 routing-mark=main
```

As can be observed, there are two variations possible - to specify gateway as *ip_address%interface* or to simply specify *interface*. The first should be used for broadcast interfaces in most cases. The second should be used for point-to-point interfaces, and also for broadcast interfaces, if the route is a connected route in some VRF. For example, if you have address `1.2.3.4/24` on interface *ether2* that is put in a VRF, there will be connected route to `1.2.3.0/24` in that VRF's routing table. It is acceptable to add static route `1.2.3.0/24` in a different routing table with interface-only gateway, even though *ether2* is a broadcast interface:

```
add dst-address=1.2.3.0/24 gateway=ether2 routing-mark=main
```

References

RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs) [1]

MPLS Fundamentals, chapter 7, *Luc De Ghein*, Cisco Press 2006

References

[1] http://www.ietf.org/rfc/rfc4364.txt

Manual:Layer-3 MPLS VPN example

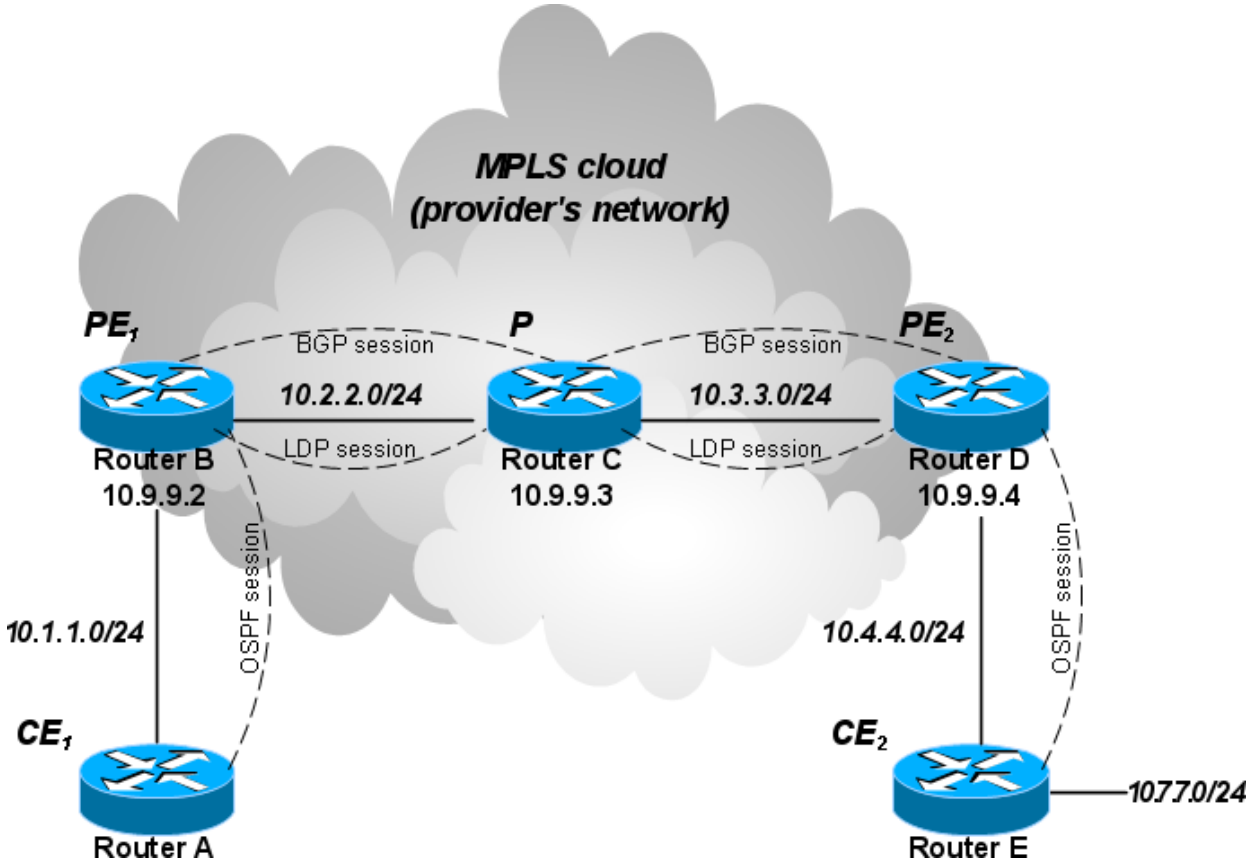
This is a kind of "putting it all together" setup.

Technologies used:

- LDP for MPLS label distribution
- BGP for VPNv4 route distribution
- OSPF as CE - PE routing protocol

Software:

- PE and P routers have RouterOS 3.17 with **routing-test** and **mpls-test** packages.
- CE routers have RouterOS 3.17 with routing-test package. (**routing** package and older versions can be used here as well.)



IP addressing & routing

Provider's network

On Router B:

```
/ip address add address=10.1.1.2/24 interface=ether2
/ip address add address=10.2.2.2/24 interface=ether3

# put PE-CE interface in a VRF
/ip route vrf add routing-mark=vrf1 interfaces=ether2 \
 route-distinguisher=10.1.1.1:111 import-route-targets=10.1.1.1:111 export-route-targets=10.1.1.1:111

# loopback interface
/interface bridge add name=lobridge
/ip address add address=10.9.9.2/32 interface=lobridge

# add routes to loopback addresses
# (static routing is used for destinations inside providers network)
/ip route add dst-address=10.9.9.3/32 gateway=10.2.2.3
/ip route add dst-address=10.9.9.4/32 gateway=10.2.2.3
```

On Router C:

```
/ip address add address=10.2.2.3/24 interface=ether3
/ip address add address=10.3.3.3/24 interface=ether2

# loopback interface
/interface bridge add name=lobridge
/ip address add address=10.9.9.3/32 interface=lobridge

# add routes to loopback addresses
/ip route add dst-address=10.9.9.2/32 gateway=10.2.2.2
/ip route add dst-address=10.9.9.4/32 gateway=10.3.3.4
```

On Router D:

```
/ip address add address=10.3.3.4/24 interface=ether2
/ip address add address=10.4.4.4/24 interface=ether3

# put PE-CE interface in a VRF
/ip route vrf add routing-mark=vrf1 interfaces=ether3 \
 route-distinguisher=10.1.1.1:111 import-route-targets=10.1.1.1:111 export-route-targets=10.1.1.1:111

# loopback interface
/interface bridge add name=lobridge
/ip address add address=10.9.9.4/32 interface=lobridge

# add routes to loopback addresses
/ip route add dst-address=10.9.9.2/32 gateway=10.3.3.3
/ip route add dst-address=10.9.9.3/32 gateway=10.3.3.3
```

Client's sites

On Router A:

```
/ip address add address=10.1.1.1/24 interface=<ToRouterB>
```

On Router E:

```
/ip address add address=10.4.4.5/24 interface=<ToRouterD>
/ip address add address=10.7.7.5/24 interface=<ToLocalNetwork>
```

LDP

On Router B:

```
/mpls ldp set enabled=yes transport-address=10.9.9.2
/mpls ldp interface add interface=ether3
```

On Router C:

```
/mpls ldp set enabled=yes transport-address=10.9.9.3
/mpls ldp interface add interface=ether2
/mpls ldp interface add interface=ether3
```

On Router D:

```
/mpls ldp set enabled=yes transport-address=10.9.9.4
/mpls ldp interface add interface=ether2
```

Setting transport address for LDP is not required, but very recommended. If the address is not set, the router will pick any address at random, which may be an address belonging to VRF, and as such not connectible from internal P routers.

Results

```
[admin@C] > /mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEN ADDRESSES
 0  O   10.9.9.2        10.9.9.3        10.1.1.2:0                no  10.1.1.2
                                                                      10.2.2.2
                                                                      10.9.9.2
 1      10.3.3.4                                                  no
 2  O   10.9.9.4        10.9.9.3        10.3.3.4:0                no  10.3.3.4
                                                                      10.4.4.4
                                                                      10.9.9.4
```

BGP

On Router B:

```
/routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes \
 redistribute-ospf=yes
/routing bgp peer add remote-address=10.9.9.3 remote-as=65530 address-families=vpnv4 \
 update-source=lobridge
```

On Router C:

```
/routing bgp peer add remote-address=10.9.9.2 remote-as=65530 route-reflect=yes \
  address-families=vpnv4 update-source=lobridge
/routing bgp peer add remote-address=10.9.9.4 remote-as=65530 route-reflect=yes \
  address-families=vpnv4 update-source=lobridge
# client-to-client-reflection is on by default
#/routing bgp instance set default client-to-client-reflection=yes
```

On Router D:

```
/routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes \
  redistribute-ospf=yes
/routing bgp peer add remote-address=10.9.9.3 remote-as=65530 address-families=vpnv4 \
  update-source=lobridge
```

Note that route reflection here is used for the sake of an example. A simpler configuration would work as well - one where there is a BGP session between B and D and C is not running BGP at all.

Results

Check for routes on PE routers:

```
/routing bgp vpn vpnv4-route print
```

and

```
/ip route print where bgp
```

OSPF

On Router A:

```
/routing ospf network add network=10.1.1.0/24 area=backbone
```

On Router B:

```
/routing ospf instance set default routing-table=vrf1 redistribute-bgp=as-type-1
/routing ospf network add network=10.1.1.0/24 area=backbone
```

On Router D:

```
/routing ospf instance set default routing-table=vrf1 redistribute-bgp=as-type-1
/routing ospf network add network=10.4.4.0/24 area=backbone
```

On Router E:

```
/routing ospf network add network=10.4.4.0/24 area=backbone
/routing ospf network add network=10.7.7.0/24 area=backbone
```

Results

Routing table on CE router A:

```
[admin@A] > /ip route pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
```

```
 0 ADC  10.1.1.0/24        10.1.1.1        ether2             0
 1 ADo  10.4.4.0/24                        10.1.1.2 reachab... 110
 2 ADo  10.7.7.0/24                        10.1.1.2 reachab... 110
```

Routing table on CE router E:

```
[admin@E] > /ip route pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADo  10.1.1.0/24                        10.4.4.4 reachab... 110
 1 ADC  10.4.4.0/24        10.4.4.5        ether2             0
 2 ADC  10.7.7.0/24        10.7.7.5        ether3             0
```

Test

On Router A:

Ping from CE1 -> to PE1:

```
[admin@A] > /ping 10.1.1.2
10.1.1.2 64 byte ping: ttl=64 time=8 ms
10.1.1.2 64 byte ping: ttl=64 time=4 ms
10.1.1.2 64 byte ping: ttl=64 time=5 ms
10.1.1.2 64 byte ping: ttl=64 time=5 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 4/5.5/8 ms
```

Ping from CE1 -> to CE2:

```
[admin@A] > /ping 10.4.4.5
10.4.4.5 64 byte ping: ttl=61 time=12 ms
10.4.4.5 64 byte ping: ttl=61 time=5 ms
10.4.4.5 64 byte ping: ttl=61 time=6 ms
10.4.4.5 64 byte ping: ttl=61 time=8 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 5/7.7/12 ms
```

[admin@A] > /ping 10.7.7.5

```
10.7.7.5 64 byte ping: ttl=61 time=14 ms
10.7.7.5 64 byte ping: ttl=61 time=4 ms
10.7.7.5 64 byte ping: ttl=61 time=8 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 4/8.6/14 ms
```

```
[admin@A] > /tool traceroute 10.7.7.5
     ADDRESS                                    STATUS
   1         10.1.1.2 3ms 6ms 2ms
   2         0.0.0.0 timeout timeout timeout
   3         10.3.3.4 4ms 3ms 3ms
   4         10.7.7.5 3ms 3ms 3ms
```

The second hop failure is normal.

To see whole MPLS cloud as one IP hop, configure **propagate-ttl=no**. This setting should be the same on all provider's routers.

On Routers B,C,D:

```
/mpls set propagate-ttl=no
```

```
[admin@A] > /tool traceroute 10.7.7.5
     ADDRESS                                          STATUS
   1         10.1.1.2 6ms 3ms 5ms
   2         10.3.3.4 5ms 3ms 6ms
   3         10.7.7.5 9ms 9ms 6ms
```

No failures here.

Connecting from PE to CE

In this case routing-table must be specified manually.

Ping from PE1 -> to CE1:

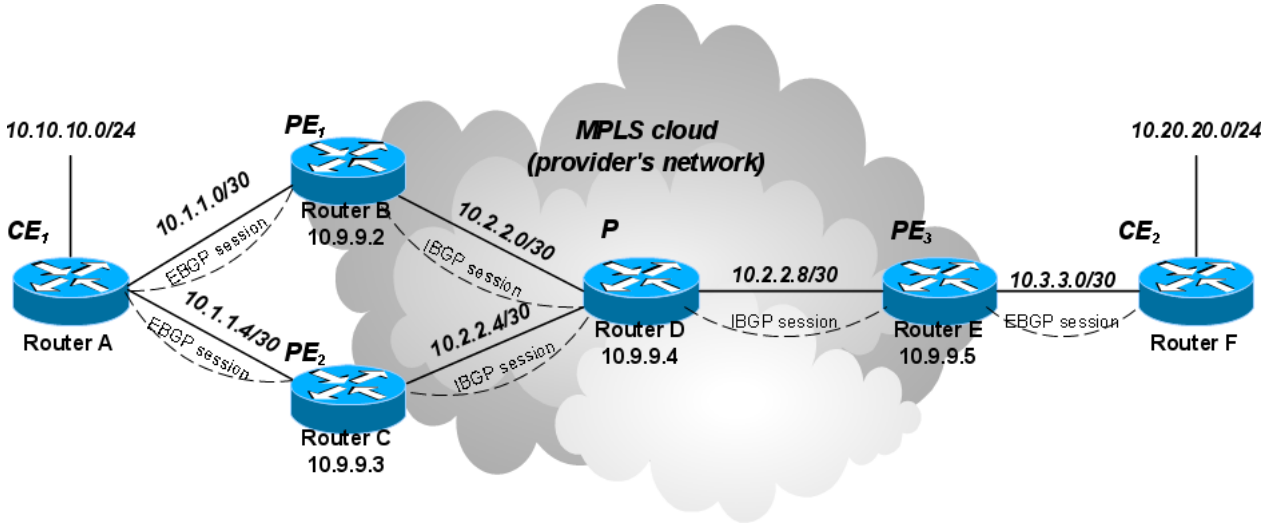
```
[admin@B] > ping 10.1.1.1 routing-table=vrf1
10.1.1.1 64 byte ping: ttl=64 time=9 ms
10.1.1.1 64 byte ping: ttl=64 time=6 ms
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 6/7.5/9 ms
```

Manual:EBGP as PE-CE routing protocol

Applies to RouterOS: v4

- *Packages required:* **routing, mpls**
- *Software versions:* **4.3+**

Setup



In this setup we describe the use of EBGP as Provider Edge - Customer Edge (PE-CE) routing protocol.

Router A and Router F both belong to the same customer's VPN, but to different sites.

Router A is multihomed - is has connections to two PEs, router B and router C.

Routers B, C, and E are PE routers.

Router D is provider (P) router and functions as BGP route reflector.

All provider's routers belong to AS 100; all customer routers belong to private AS 65000.

Description

There are several tricky aspects about this setup.

First, it is not possible to use BGP built-in mechanism of routing loop prevention, that checks BGP AS path for presence of local AS path numbers and discards all routes that match. We want to distribute routes from A to F, and vice versa, but they belong to the same BGP AS. (One solution is to use different private AS numbers there, but that's not always possible or desirable.)

- One way to do work around this BGP AS path loop check is to configure BGP **as-override** option at exit point from provider's network.
- Another way is to configure **remove-private-as** at providers network entry point (it will work only if customer's AS numbers are private, of course!)
- Yet another way is to configure **allow-as-in=x** on customers edge router. "x" is the number of times local as number can be present in AS path.

In this configuration we use the **as-override** option on router E (to make router F accept routes from A), and **allow-as-in** option on router A, to make it accept routes from F.

Router A:

```
routing bgp peer add remote-address=10.1.1.2 remote-as=100 allow-as-in=1;
routing bgp peer add remote-address=10.1.1.6 remote-as=100 allow-as-in=1;
```

Router E:

```
routing bgp peer add instance=ebgp remote-address=10.3.3.2 remote-as=65000 as-override=yes;
```

The second tricky aspect is that since CE1 is multihomed (i.e. has links to multiple PEs) and BGP AS path loop prevention mechanism is disabled on router A because 'allow-as-in' option configured, the routes that A advertises to one PE router may be received back from the second PE. Installing those route in VRF table can also lead to suboptimal routing and even to BGP convergence failure. To avoid that, BGP Site of Origin (SOO) extended communities can be used. In this configuration we configure routing filter on PE routers that sets BGP SOO extended communities to routes received from CE router, and another filter, that filters out VPNv4 routes received from IBGP by the same SOO extended community attribute.

Routers B, C:

```
routing filter add chain=ibgp-in site-of-origin=1:100 action=discard;
routing filter add chain=ebgp-in set-site-of-origin=1:100;
```

We also use different BGP instances on PE routers: one for PE-CE (i.e. EBGP) peers and one for provider's network internal BGP peers.

Configuration

Router A:

```
 ip address add address=10.1.1.1/30 interface=A_B;
 ip address add address=10.1.1.5/30 interface=A_C;
 interface bridge add name=somenet;
 ip address add address=10.10.10.1/24 interface=somenet;
 routing bgp instance set default as=65000 redistribute-connected=yes;
 routing bgp peer add remote-address=10.1.1.2 remote-as=100 allow-as-in=1;
 routing bgp peer add remote-address=10.1.1.6 remote-as=100 allow-as-in=1;
```

Router B:

```
ip address add address=10.1.1.2/30 interface=B_A;
ip address add address=10.2.2.1/30 interface=B_D;
interface bridge add name=lobridge;
ip address add address=10.9.9.2/32 interface=lobridge;
ip route add dst-address=10.9.9.3 gateway=10.2.2.2;
ip route add dst-address=10.9.9.4 gateway=10.2.2.2;
ip route add dst-address=10.9.9.5 gateway=10.2.2.2;
ip route vrf add routing-mark=vrf1 interfaces=B_A route-distinguisher=1:1 \
 import-route-targets=1:1 export-route-targets=1:1;
mpls ldp set enabled=yes transport-address=10.9.9.2;
mpls ldp interface add interface=B_D hello-interval=3;
routing bgp instance set default as=100;
routing bgp instance add name=ebgp router-id=0.0.0.2 as=100 routing-table=vrf1;
routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes \
 redistribute-other-bgp=yes;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.4 remote-as=100 \
 in-filter=ibgp-in out-filter=ibgp-out update-source=10.9.9.2;
routing bgp peer add instance=ebgp remote-address=10.1.1.1 remote-as=65000 \
  in-filter=ebgp-in out-filter=ebgp-out;
routing filter add chain=ebgp-out site-of-origin=1:100 action=discard;
routing filter add chain=ebgp-in set-site-of-origin=1:100;
```

Router C:

```
ip address add address=10.1.1.6/30 interface=C_A;
ip address add address=10.2.2.5/30 interface=C_D;
interface bridge add name=lobridge;
ip address add address=10.9.9.3/32 interface=lobridge;
ip route add dst-address=10.9.9.2 gateway=10.2.2.6;
ip route add dst-address=10.9.9.4 gateway=10.2.2.6;
ip route add dst-address=10.9.9.5 gateway=10.2.2.6;
ip route vrf add routing-mark=vrf1 interfaces=C_A route-distinguisher=1:1 \
 import-route-targets=1:1 export-route-targets=1:1;
mpls ldp set enabled=yes transport-address=10.9.9.3;
mpls ldp interface add interface=C_D hello-interval=3;
routing bgp instance set default as=100;
routing bgp instance add name=ebgp router-id=0.0.0.3 as=100 routing-table=vrf1;
```

```
routing bgp instance vrf add instance=default routing-mark=vrf1 \
 redistribute-connected=yes redistribute-other-bgp=yes;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.4 remote-as=100 \
 in-filter=ibgp-in update-source=10.9.9.3;
routing bgp peer add instance=ebgp remote-address=10.1.1.5 remote-as=65000 \
 in-filter=ebgp-in out-filter=ebgp-out;
routing filter add chain=ibgp-in site-of-origin=1:100 action=discard;
routing filter add chain=ebgp-in set-site-of-origin=1:100;
```

Router D:

```
ip address add address=10.2.2.2/30 interface=D_B;
ip address add address=10.2.2.6/30 interface=D_C;
ip address add address=10.2.2.9/30 interface=D_E;
interface bridge add name=lobridge;
ip address add address=10.9.9.4/32 interface=lobridge;
ip route add dst-address=10.9.9.2 gateway=10.2.2.1;
ip route add dst-address=10.9.9.3 gateway=10.2.2.5;
ip route add dst-address=10.9.9.5 gateway=10.2.2.10;
mpls ldp set enabled=yes transport-address=10.9.9.4;
mpls ldp interface add interface=D_B hello-interval=3;
mpls ldp interface add interface=D_C hello-interval=3;
mpls ldp interface add interface=D_E hello-interval=3;
routing bgp instance set default as=100;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.2 remote-as=100 \
 update-source=10.9.9.4 route-reflect=yes;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.3 remote-as=100 \
 update-source=10.9.9.4 route-reflect=yes;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.5 remote-as=100 \
 update-source=10.9.9.4 route-reflect=yes;
```

Router E:

```
 ip address add address=10.3.3.1/30 interface=E_F;
 ip address add address=10.2.2.10/30 interface=E_D;
 interface bridge add name=lobridge;
 ip address add address=10.9.9.5/32 interface=lobridge;
 ip route add dst-address=10.9.9.2 gateway=10.2.2.9;
 ip route add dst-address=10.9.9.3 gateway=10.2.2.9;
 ip route add dst-address=10.9.9.4 gateway=10.2.2.9;
 ip route vrf add routing-mark=vrf1 interfaces=E_F route-distinguisher=1:1 \
  import-route-targets=1:1 export-route-targets=1:1;
 mpls ldp set enabled=yes transport-address=10.9.9.5;
 mpls ldp interface add interface=E_D hello-interval=3;
 routing bgp instance set default as=100;
 routing bgp instance add name=ebgp router-id=0.0.0.5 as=100 routing-table=vrf1;
 routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes \
  redistribute-other-bgp=yes;
 routing bgp peer add address-families=vpnv4 remote-address=10.9.9.4 remote-as=100 \
```

```
  update-source=10.9.9.5;
 routing bgp peer add instance=ebgp remote-address=10.3.3.2 remote-as=65000 as-override=yes;
```

Router F:

```
ip address add address=10.3.3.2/30 interface=F_E;
interface bridge add name=somenet;
ip address add address=10.20.20.1/24 interface=somenet;
routing bgp instance set default as=65000 redistribute-connected=yes;
routing bgp peer add remote-address=10.3.3.1 remote-as=100;
```

Results

Routes on CE1 router A:

```
 [admin@A] > ip route print detail
 Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp,
 o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

1 ADC dst-address=10.1.1.4/30 pref-src=10.1.1.5 gateway=A_C gateway-status=A_C reachable
 distance=0 scope=10

2 ADb dst-address=10.3.3.0/30 gateway=10.1.1.2 gateway-status=10.1.1.2 reachable A_B
 distance=20 scope=40 target-scope=10 bgp-as-path=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer1

3  Db dst-address=10.3.3.0/30 gateway=10.1.1.6 gateway-status=10.1.1.6 reachable A_C
 distance=20 scope=40 target-scope=10 bgp-as-path=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer2

4 ADC dst-address=10.10.10.1/30 pref-src=10.1.1.1 gateway=somenet
 gateway-status=somenet reachable distance=0 scope=10

5 ADb dst-address=10.20.20.0/24 gateway=10.1.1.2 gateway-status=10.1.1.2 reachable A_B
 distance=20 scope=40 target-scope=10 bgp-as-path=100,65000 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer1

6  Db dst-address=10.20.20.0/24 gateway=10.1.1.6 gateway-status=10.1.1.6 reachable A_C
 distance=20 scope=40 target-scope=10 bgp-as-path=100,65000 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer2
```

Routes on CE2 router F:

```
 [admin@F] > ip route print detail
 Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp,
 o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

0 ADb dst-address=10.1.1.0/30 gateway=10.3.3.1 gateway-status=10.3.3.1 reachable F_E
 distance=20 scope=40 target-scope=10 bgp-as-path=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer1
```

```
1 ADb dst-address=10.1.1.4/30 gateway=10.3.3.1 gateway-status=10.3.3.1 reachable F_E
 distance=20 scope=40 target-scope=10 bgp-as-path=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer1

2 ADC dst-address=10.3.3.0/30 pref-src=10.3.3.2 gateway=F_E gateway-status=F_E reachable
 distance=0 scope=10

3 ADb dst-address=10.10.10.0/24 gateway=10.3.3.1 gateway-status=10.3.3.1 reachable F_E
 distance=20 scope=40 target-scope=10 bgp-as-path=100,100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1,SOO:1:100 received-from=peer1

4 ADC dst-address=10.20.20.0/30 pref-src=10.20.20.1 gateway=somenet
 gateway-status=somenet reachable distance=0 scope=10
```

Routes on PE1 router B:

```
 [admin@B] > ip route print detail
 Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp,
 o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

0 ADC dst-address=10.1.1.0/30 pref-src=10.1.1.2 gateway=B_A gateway-status=B_A reachable
 distance=0 scope=10 routing-mark=vrf1

1  Db dst-address=10.1.1.0/30 gateway=10.1.1.1 gateway-status=10.1.1.1 on vrf1 reachable A_B
 distance=20 scope=40 target-scope=10 routing-mark=vrf1 bgp-as-path=65000
 bgp-origin=incomplete bgp-ext-communities=SOO:1:100 received-from=peer2

2 ADb dst-address=10.1.1.4/30 =gateway=10.1.1.1 gateway-status=10.1.1.1 on vrf1 reachable B_A
 distance=20 scope=40 target-scope=10 routing-mark=vrf1 bgp-as-path=65000
 bgp-origin=incomplete bgp-ext-communities=SOO:1:100 received-from=peer2

3  Db dst-address=10.1.1.4/30 gateway=10.9.9.3 gateway-status=10.9.9.3 recursive via 10.2.2.2 B_D
 distance=20 scope=40 target-scope=30 routing-mark=vrf1 bgp-local-pref=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1

4 ADb dst-address=10.3.3.0/30 gateway=10.9.9.5 gateway-status=10.9.9.5 recursive via 10.2.2.2 B_D
 distance=20 scope=40 target-scope=30 routing-mark=vrf1 bgp-local-pref=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1

5 ADb dst-address=10.10.10.0/24 gateway=10.1.1.1 gateway-status=10.1.1.1 on vrf1 reachable B_A
 distance=20 scope=40 target-scope=10 routing-mark=vrf1 bgp-as-path=65000 bgp-origin=incomplete
 bgp-ext-communities=SOO:1:100 received-from=peer2

6 ADb dst-address=10.20.20.0/24 gateway=10.9.9.5 gateway-status=10.9.9.5 recursive via 10.2.2.2 B_D
 distance=20 scope=40 target-scope=30 routing-mark=vrf1 bgp-as-path=65000 bgp-local-pref=100
 bgp-origin=incomplete bgp-ext-communities=RT:1:1

7 ADC dst-address=10.2.2.0/30 pref-src=10.2.2.1 gateway=B_D gateway-status=B_D reachable
```

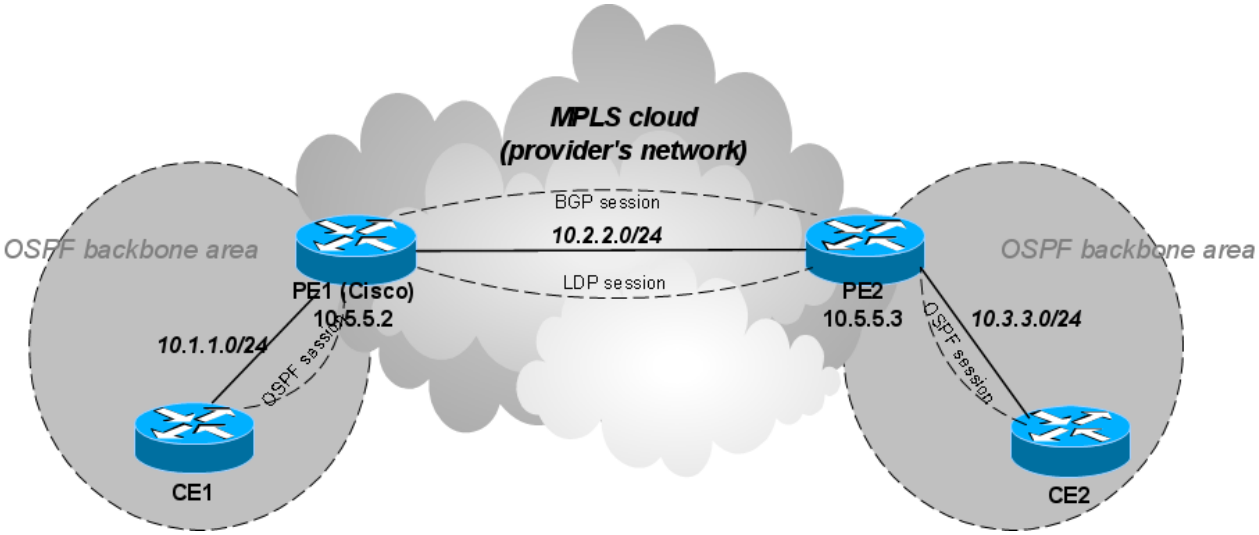
```
 distance=0 scope=10

8 ADC dst-address=10.9.9.2/32 pref-src=10.9.9.2 gateway=lobridge gateway-status=lobridge reachable
 distance=0 scope=10

9 A S dst-address=10.9.9.3/32 gateway=10.2.2.2 gateway-status=10.2.2.2 reachable B_D
 distance=1 scope=30 target-scope=10

10 A S dst-address=10.9.9.4/32 gateway=10.2.2.2 gateway-status=10.2.2.2 reachable B_D
 distance=1 scope=30 target-scope=10

11 A S dst-address=10.9.9.5/32 gateway=10.2.2.2 gateway-status=10.2.2.2 reachable B_D
 distance=1 scope=30 target-scope=10
```

Manual:OSPF as PE-CE routing protocol

Software:

- PE1 router is Cisco 7200
- PE2 is MT and has RouterOS 3.23 with **routing-test** and **mpls-test** packages.
- CE1 and CE2 have any RouterOS version.

Configuration with inter-area routing



CE1

```
/ip address add address=10.1.1.1/24 interface=ether1
# static route to redistribute
/ip route add dst-address=10.10.1.0/24 gateway=x.x.x.x
/routing ospf instance set default redistribute-static=as-type-1 router-id=0.0.0.1
/routing ospf network add area=backbone network=1.1.1.0/24
```

CE2

```
/ip address add address=10.3.3.4/24 interface=ether1
# static route to redistribute
/ip route add dst-address=10.10.4.0/24 gateway=y.y.y.y
/routing ospf instance set default redistribute-static=as-type-1 router-id=0.0.0.4
/routing ospf network add area=backbone network=10.3.3.0/24
```

PE1 (Cisco)

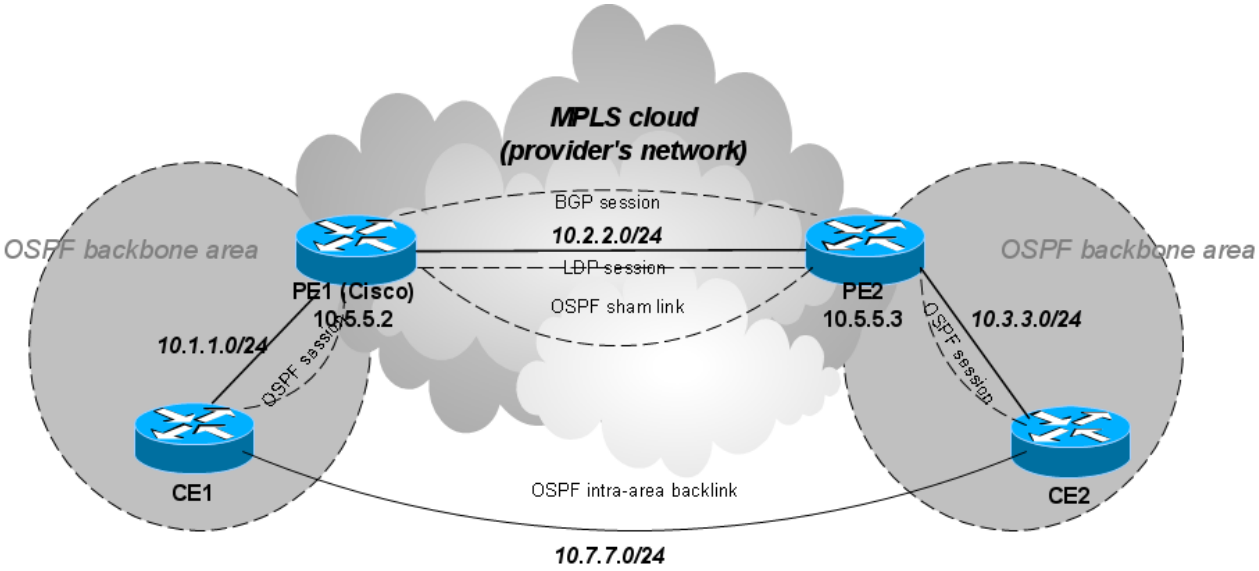
```
ip vrf vrf1
 rd 1.1.1.1:111
 route-target export 1.1.1.1:111
 route-target import 1.1.1.1:111
 exit

interface Loopback0
 ip address 10.5.5.2 255.255.255.255

mpls ldp router-id Loopback0 force
mpls label protocol ldp

interface FastEthernet0/0
 ip vrf forwarding vrf1
 ip address 10.1.1.2 255.255.255.0

interface FastEthernet1/0
 ip address 10.2.2.2 255.255.255.0
 mpls ip

router ospf 1 vrf vrf1
 router-id 10.5.5.2
 network 10.1.1.0 0.0.0.255 area 0
 redistribute bgp 65000 subnets
 domain-id 0.0.0.1
 domain-tag 2222

router bgp 65000
 neighbor 10.5.5.3 remote-as 65000
 neighbor 10.5.5.3 update-source Loopback0
 address-family vpnv4
  neighbor 10.5.5.3 activate
  neighbor 10.5.5.3 send-community both
  exit-address-family
 address-family ipv4 vrf vrf1
  redistribute connected
  redistribute ospf 1 vrf vrf1 match internal external
  exit-address-family
```

```
ip route 10.5.5.3 255.255.255.255 10.2.2.3
```

PE2

```
/interface bridge add name=lobridge

/ip address
 add address=10.2.2.3/24 interface=ether1
 add address=10.3.3.3/24 interface=ether2
 add address=10.5.5.3/32 interface=lobridge

/ip route vrf add disabled=no export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111 \
    interfaces=ether2,vrf-lobridge route-distinguisher=1.1.1.1:111 routing-mark=vrf1
/ip route add dst-address=10.5.5.2/32 gateway=10.2.2.2

/routing bgp instance set default as=65000
/routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes redistribute-ospf=yes
/routing bgp peer add instance=default remote-as=65000 remote-address=10.5.5.2 \
       address-families=vpnv4 update-source=lobridge

/routing ospf instance redistribute-bgp=as-type-1 router-id=10.5.5.3 routing-table=vrf1 \
       domain-id=0.0.0.1 domain-tag=3333
/routing ospf network add area=backbone network=10.3.3.0/24

/mpls ldp set enabled=yes transport-address=10.5.5.3
/mpls ldp interface add interface=ether1
```

Configuration with intra-area routing (including a sham link)



CE1 additional backlink

```
/ip address add address=10.7.7.1/24 interface=backlink
/routing ospf network add area=backbone network=10.7.7.0/24
/routing ospf interface add interface=backlink cost=1000 network-type=point-to-point
```

CE2 additional backlink

```
/ip address add address=10.7.7.4/24 interface=backlink
/routing ospf network add area=backbone network=10.7.7.0/24
/routing ospf interface add interface=backlink cost=1000 network-type=point-to-point
```

PE1 (Cisco) with a sham link

```
interface Loopback1
 ip vrf forwarding vrf1
 ip address 10.6.6.2 255.255.255.255

router ospf 1 vrf vrf1
 area 0 sham-link 10.6.6.2 10.6.6.3 cost 10

ip route 10.6.6.3 255.255.255.255 10.2.2.3

! all the rest of settings remain unchanged
```

PE2 with a sham-link

```
/interface bridge add name=vrf-lobridge

/ip address add address=10.6.6.3/32 interface=vrf-lobridge
# change the VRF to include vrf-lobridge interface
/ip route vrf add disabled=no export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111 \
    interfaces=ether2,vrf-lobridge route-distinguisher=1.1.1.1:111 routing-mark=vrf1
# configure the sham link
/routing ospf sham-link add area=backbone src-address=10.6.6.3 dst-address=10.6.6.2
# add route to sham link's remote address
/ip route add dst-address=10.6.6.2 gateway=10.2.2.2
```

Manual:TE Tunnels

Overview

For MPLS overview and RouterOS supported MPLS features see MPLS Overview.

MPLS RSVP TE tunnels are a way to establish unidirectional label switching paths. In general RSVP TE serves similar purpose as label distribution using LDP protocol - establishing label switched path that ensures frame delivery from ingress to egress router, but with additional features:

- possibility to establish label switching path using either full or partial explicit route;
- constraint based LSP establishment - label switching path is established over links that fulfill requirements, such as bandwidth and link properties.

MPLS RSVP TE is based on RSVP protocol with extensions introduced by RFC 3209 that adds support for explicit route and label exchange.

Note that constraints for path establishment are purely controlled by administrator - for example, bandwidth of link participating in RSVP TE network is set by administrator and does not necessarily reflect real bandwidth of the link. The same way bandwidth reserved for tunnel is set by administrator and does not automatically imply any limits on traffic sent over tunnel. Therefore at any moment in time, bandwidth available on TE link is bandwidth configured for link minus sum of all reservations made on link, not physically available bandwidth which can be either less (in case data is forwarded over tunnels with rate that exceeds bandwidth reserved for tunnel or if non-RSVP tunnel data is forwarded over link as well) or more (in case data is forwarded over tunnels with rate smaller than allocated for tunnel) than bandwidth available for reservations.

RSVP TE tunnels are initiated by head-end (ingress router) of tunnel. Head-end router sends RSVP Path message containing necessary parameters towards tail-end of the tunnel. Routers along the path ensure that they can forward Path message towards next hop, taking into acount path constraints. Once Path message reaches tail-end of the tunnel, tail-end router sends RSVP Resv message in the opposite direction. Resv message hop by hop traverses exactly the same path that Path message, only in the opposite direction. Each router forwarding Resv message allocates necessary bandwith on appropriate downstream link if possible. Once head-end router succesfully receives Resv message that matches sent Path message, tunnel can be considered established. Tunnel is maintained by periodically refreshing its state using Path and Resv messages.

RSVP TE tunnels can be established with number of path options:

- along path that data from head-end of tunnel is routed to tail-end - in this case each router along tunnel path figures out next hop of tunnel based on routing table. If at some point usable route is not found or downstream interface does not meet constraints (for example if requested bandwidth exceeds available bandwidth), tunnel can not be established.
- along statically configured explicit path - in this case each router along tunnel path figures out next hop of tunnel based on explicit route specified in Path message. This explicit route can be either complete (specifies all routers along the path in the order they must be traversed) or partial (specifies only some routers that must be traversed). To decide next hop router, each router along the path look up route to next router specified in explicit route. If no usable route is found or downstream interface does not meet constraints, tunnel can not be established
- Constrained Shortest Path First - in this case head-end router calculates path to tail-end using its knowledge of network state - properties of links and available bandwidth. This option needs assistance from IGP routing protocol (such as OSPF) to distribute bandwidth information throughout the network. This is implemented in OSPF by means of opaque LSAs. When using CSPF, head-end router calculates path that satisfies the requirements and produces explicit path for Path message. If path that matches constraints can not be calculated, tunnel can not be established. Dynamically calculated path can also be partially explicit - in this case CSPF seeks

for shortest path matching constraints between every two explicit hops. If explicit path is specified completely and CSPF is used, CSPF just checks if this path meets the constraints taking into account knowledge about link states in network - so instead of failure to establish tunnel while forwarding Path message in network, Path message is not even sent as it is clear that establishing tunnel will fail.

Forwarding traffic onto TE tunnels

RSVP TE tunnel head-end appears as interface in RouterOS. Note that RSVP TE tunnels are unidirectional - it is not necessary to have matching tunnel for reverse direction on tail-end router. When tail-end router receives data sent over tunnel, it either receives it with TE tunnel label stripped off by penultimate hop (non-default behaviour) or with explicit-null label, which gets stripped and packet is further inspected (if tunnel label is last label in stack, packet gets routed, otherwise it is processed based on next label in stack, for example, as VPLS packet). Bidirectional tunnel can be simulated by creating one tunnel in one direction and other in other direction between the same endpoints. Still no data will be accounted as received over TE tunnel, as in reality both tunnels are unrelated.

One way to forward traffic onto tunnel is to use routing, but this limits TE tunnel to be used only for routing IP packets.

Additionally, several types of traffic can be forwarded onto TE tunnel automatically, if it is known to be destined to the endpoint of tunnel and if tunnel is active:

- traffic that is routed using route route learned from BGP, if BGP NextHop is tunnel endpoint (this default behaviour can be changed by setting route porperty "use-te-nexthop" to "no"), both - regular IP and VPNv4 (MP-BGP IP VPN) routes fit in this category;
- traffic for VPLS interfaces, if remote endpoint of VPLS pseudowire is the same as TE tunnel endpoint.

For example, for IP BGP route having BGP NextHop x.x.x.x, forwarding method will be chosen according to the following rules:

- if TE tunnel with endpoint x.x.x.x is active, use it;
- otherwise if LDP label mapping from next hop towards x.x.x.x is received, use it;
- otherwise use regular routing (no MPLS encapsulation).

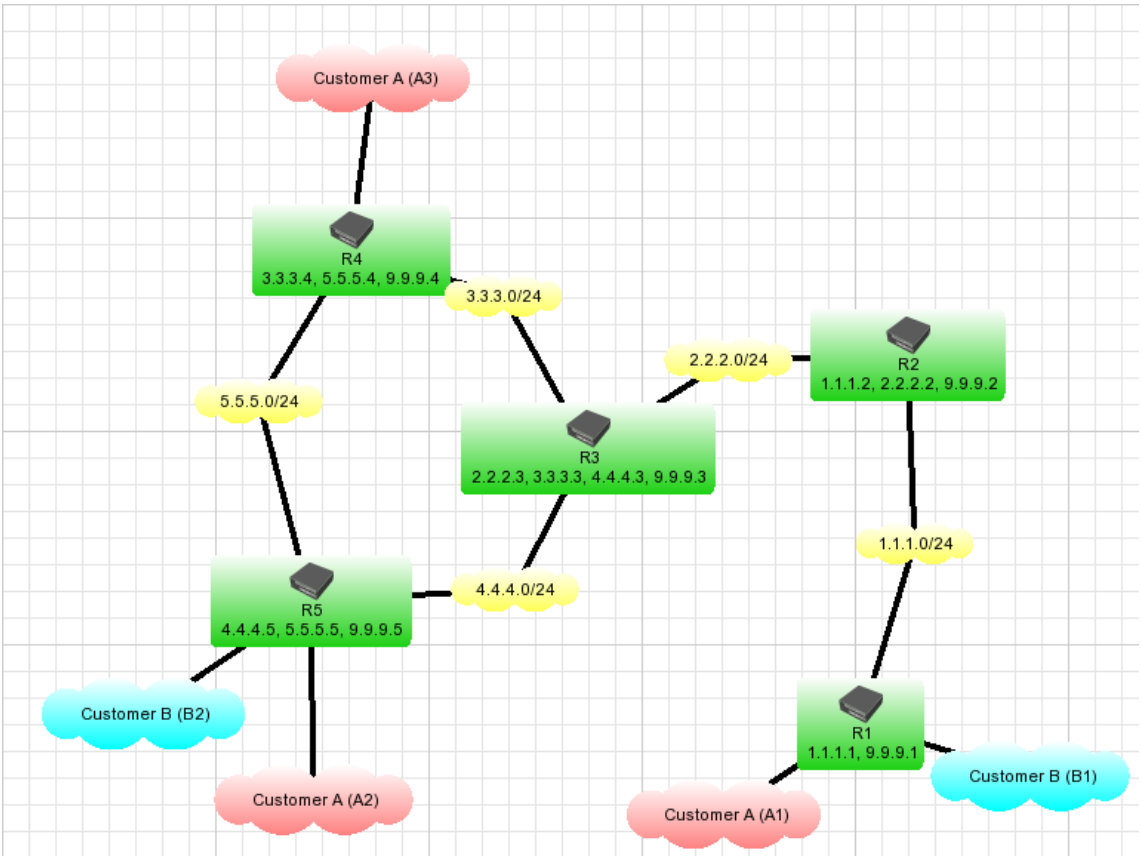
In similar way, if remote address of VPLS pseudowire is x.x.x.x, forwarding method will be chosen in the following order:

- if TE tunnel with endpoint x.x.x.x is active, use it;
- otherwise if LDP label mapping from next hop towards x.x.x.x is received, use it;
- otherwise VPLS tunnel can not be active.

Note that RSVP TE tunnels as a way to establish LSPs can be used together with LDP. Using RSVP TE does not replace or disable LDP, but LSP established by TE is usually preferred over one established using LDP.

Example network

Consider the same network as used for LDP signaled VPLS example in MPLSVPLS:



Customer A wants to establish IP VPN between his 3 sites and Customer B wants to transparent connection for ethernet segments at his sites.

Prerequisites for MPLS TE

In general, prerequisites for using MPLS TE are the same as mentioned in MPLSVPLS, but there are a few details:

- by default TE tunnel tail-end router advertises explicit null label, therefore penultimate hop popping does not happen (the purpose of using explicit null label is to communicate QoS information in MPLS label Exp field), so main purpose of having "loopback" IP address for every router is to have tunnel endpoints unaffected by link state changes;
- in order to use CSPF path selection for tunnels, OSPF must be configured and running in network.

Enabling TE support

In order for OSPF to distribute TE information, TE related OSPF parameters must be set:

```
[admin@R1] > /routing ospf set mpls-te-area=backbone mpls-te-router-id=lobridge
```

This instructs OSPF to distribute TE information in "backbone" area using IP address of "lobridge" as router ID.

In order for router to be able to participate in TE tunnel (either as head-end, tail-end or forwarding router), TE support must be enabled. TE support must be enabled on all interfaces that will receive and send RSVP TE protocol packets. On R1 it is done by commands (interface ether3 is facing network 1.1.1.0/24):

```
[admin@R1] > /mpls traffic-eng interface add interface=ether3 bandwidth=100000
```

This configures ether3 interface with TE support, having bandwidth 100000 Bps. Other routers are configured in similar way.

As soon as TE support is enabled on interface, appropriate opaque LSAs are distributed into OSPF area. For example, on R1 it can be seen, that there is total 15 opaque LSAs in LSA database:

```
[admin@R1] > /routing ospf lsa print
  ...
backbone              opaque-area  1.0.0.0        1.1.1.2        0x80000004      1038
backbone              opaque-area  1.0.0.0        2.2.2.3        0x80000004      1039
backbone              opaque-area  1.0.0.0        3.3.3.4        0x80000004      1038
backbone              opaque-area  1.0.0.0        4.4.4.5        0x80000004      1038
backbone              opaque-area  1.0.0.0        11.11.11.1     0x80000004      1037
backbone              opaque-area  1.0.0.1        1.1.1.2        0x80000004      1038
backbone              opaque-area  1.0.0.1        2.2.2.3        0x80000004      1039
backbone              opaque-area  1.0.0.1        3.3.3.4        0x80000004      1037
backbone              opaque-area  1.0.0.1        4.4.4.5        0x80000004      1038
backbone              opaque-area  1.0.0.2        1.1.1.2        0x80000004      1038
backbone              opaque-area  1.0.0.2        2.2.2.3        0x80000004      1039
backbone              opaque-area  1.0.0.2        3.3.3.4        0x80000004      1037
backbone              opaque-area  1.0.0.2        4.4.4.5        0x80000004      1038
backbone              opaque-area  1.0.0.3        2.2.2.3        0x80000004      1039
backbone              opaque-area  1.0.0.3        11.11.11.1     0x80000004      1037
  ...
```

Creating basic TE tunnel

Assume that we want to create TE tunnel from R1 to R5. In order to do this, tunnel path specification must be created:

```
[admin@R1] > /mpls traffic-eng tunnel-path add use-cspf=yes name=dyn
```

This creates path template for purely dynamic path that will use CSPF.

Next, TE tunnel itself must be created:

```
[admin@R1] /interface traffic-eng> add name=te1 bandwidth=1000 primary-path=dyn \
   from-address=9.9.9.1 to-address=9.9.9.5 disabled=no record-route=yes
```

We can monitor tunnel to see its state:

```
[admin@R1] /interface traffic-eng> monitor 0
             tunnel-id: 7
    primary-path-state: established
          primary-path: dyn
  secondary-path-state: not-necessary
           active-path: dyn
          active-lspid: 1
          active-label: 29
        explicit-route: "S:1.1.1.2/32,S:2.2.2.2/32,S:2.2.2.3/32,S:4.4.4.3/32,S:4.4.4.5/32"
        recorded-route: "1.1.1.2[30],2.2.2.3[29],4.4.4.5[0]"
```

Notice, that CSPF has created explicit route that traverses R2, R3 and R5 (tail-end). TE tunnel was requested to record route it is traversing (by "record-route=yes" setting), recorded route is displayed in status along with labels that particular router has allocated for this tunnel.

Once TE tunnel is established, VPLS interface from R1 to R5 automatically switches to use this TE tunnel:

```
[admin@R1] /interface vpls> monitor 0
       remote-label: 24
        local-label: 25
      remote-status:
          transport: te1
  transport-nexthop: 1.1.1.2
     imposed-labels: 30,24
```

On routers in between R1 and R5, RSVP path and reservation state can be monitored, for example on R2:

```
[admin@R2] > /mpls traffic-eng path-state print
Flags: L - locally-originated, E - egress, F - forwarding, P - sending-path, R - sending-resv
 #      SRC             DST             BANDWIDTH  OUT-INTERFACE                        OUT-NEXT-HOP
 0  FPR 9.9.9.1:1       9.9.9.5:2       1000       ether2                               2.2.2.3
[admin@R2] > /mpls traffic-eng resv-state print
Flags: E - egress, A - active, N - non-output, S - shared
 #    SRC             DST             BANDWIDTH  LABEL               INTERFACE         NEXT-HOP
 0 AS 9.9.9.1:1       9.9.9.5:7       1000       30                  ether2            2.2.2.3
```

Note, that available bandwidth on ether2 interface (connected to R3) on R2 has changed:

```
[admin@R2] > /mpls traffic-eng interface print
Flags: X - disabled, I - invalid
 #   INTERFACE                                    BANDWIDTH  TE-METRIC  REMAINING-BW
 0   ether1                                       100000     1          100000
 1   ether2                                       100000     1          99000
```

Manual:TE tunnel auto bandwidth

Overview

By default MPLS TE tunnels do not apply any rate limitation on traffic that gets sent over tunnel. That way "bandwidth" settings for MPLS TE enabled interfaces and TE tunnels are only used for reservation accounting. There are also no means to adjust bandwidth that gets reserved for tunnel other than changing tunnel configuration no matter what is actual amount of traffic sent over tunnel. To make TE tunnels more flexible and easy to use, the following features have been introduced:

- Bandwidth limitation
- Automatic bandwidth adjustment

These features operate on tunnel head end (ingress) router. These features can either be used alone or in combination.

Bandwidth limitation

TE tunnel can be configured to limit the rate at which traffic is allowed to enter the tunnel. Limit is specified on ingress router in percent of tunnel bandwidth. E.g. creating the following tunnel:

```
[admin@R1] /interface traffic-eng> add name=te1 from-address=9.9.9.1 to-address=9.9.9.5 \
   bandwidth=100000 bandwidth-limit=120 primary-path=stat
```

means that tunnel will reserve bandwidth of 100 kilobits per second across MPLS backbone from 9.9.9.1 to 9.9.9.5 and that ingress router will limit the rate of traffic entering the tunnel to 120 kilobits per second (120% of 100 kilobits per second bandwidth). This can be confirmed by monitoring tunnel interface:

```
[admin@R1] /interface traffic-eng> monitor te1
              tunnel-id: 3
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 1
           active-label: 20
     reserved-bandwidth: 100.0kbps
             rate-limit: 120.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

Note that by default any limiting is disabled. By specifying limit as percentage of tunnel bandwidth, TE tunnel bandwith limits can be configured in rather flexible ways - some tunnels can be configured to hard limit while others can be configured with reasonable reserve, achieving different classes of service.

Automatic bandwidth adjustment

Auto bandwidth adjustment feature enables MPLS TE network to follow the changes of amount of data transmitted over tunnel. Bandwidth adjustment feature works as follows:

- Actual amount of data entering tunnel during averaging interval (**auto-bandwidth-avg-interval**) is measured, producing average rate.
- Tunnel keeps track of highest average rate seen during update interval (**auto-bandwidth-update-interval**)
- When update interval expires, TE tunnel bandwidth is updated to highest observed average rate, taking into account specified range over which bandwidth is allowed to change (**auto-bandwidth-range**)

Auto bandwidth adjustment feature gets enabled by specifying **auto-bandwidth-range**. For example, adding the following tunnel:

```
[admin@R1] /interface traffic-eng> add name=te1 from-address=9.9.9.1 to-address=9.9.9.5 \
  bandwidth=100000 primary-path=stat auto-bandwidth-range=10000-500000 \
  auto-bandwidth-avg-interval=10s auto-bandwidth-update-interval=1m
```

means that tunnel will measure average rate over 10 second periods and once per minute will update bandwidth in range from 10 to 500 kilobits per second. Tunnel **bandwidth** setting specifies the initial bandwidth of tunnel. The above tunnel in complete absence of data over it after 1 minute will change its bandwidth to specified minimum 10 kbps:

```
[admin@R1] /interface traffic-eng> monitor te1
               tunnel-id: 3
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
           active-label: 21
     reserved-bandwidth: 10.0kbps
             rate-limit: 12.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

Additionally, tunnel can be configured to reserve more bandwidth than measured. This can be achieved with **auto-bandwidth-reserve** setting which specifies percentage of additional bandwidth to reserve - so setting **auto-bandwith-reserve** to 10 means that tunnel will reserve 10% more bandwidth than measured (but will still obey the **auto-bandwidth-range**). For example changing above tunnel and running constant stream of 50kbps through it will yield the following results:

```
[admin@R1] /interface traffic-eng> set te1 auto-bandwidth-reserve=30
```

In the beginning tunnel reserves its initially specified bandwidth:

```
[admin@R1] /interface traffic-eng> monitor te1
               tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 1
```

```
           active-label: 27
     reserved-bandwidth: 100.0kbps
              rate-limit: 120.0kbps
     rate-measured-last: 48.8kbps
  rate-measured-highest: 48.8kbps
```

After update period and after previous reservations are torn down notice how reserved bandwidth exceeds average rate by 30%. Also notice that **rate-limit** correctly changes to 120% of **reserved-bandwidth**:

```
[admin@R1] /interface traffic-eng> monitor te1
               tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
           active-label: 28
     reserved-bandwidth: 64.4kbps
              rate-limit: 77.3kbps
     rate-measured-last: 48.8kbps
  rate-measured-highest: 48.8kbps
```

Note that in case reservation must be updated to lower value, brief period after update period **reserved-bandwidth** will still display previous reservation value. The reason for this is that new reservation is made without disrupting the previous tunnel and therefore shares its reservation until old reservation is torn down. **rate-limit** on turn is correctly updated to intended value. In the above example, after stopping the 50kbps stream and after update period will pass with tunnel being idle, for a brief period after update tunnel info can be:

```
[admin@R1] /interface traffic-eng> monitor te1
               tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
           active-label: 34
     reserved-bandwidth: 63.4kbps
              rate-limit: 12.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

After previous reservation (63.4kbps) is torn down, **reserved-bandwidth** correctly changes to 10kbps:

```
[admin@R1] /interface traffic-eng> monitor 1
               tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
```

```
           active-label: 34
     reserved-bandwidth: 10.0kbps
             rate-limit: 12.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

Note that **auto-bandwidth-reserve** is applied to actual measured bandwidth, before range checking according to **auto-bandwidth-range** - therefore 10kbps gets reserved, instead of 13kbps.

Combining bandwidth limitation with automatic bandwidth adjustment

Auto bandwidth adjustment can be used in combination with bandwidth limit feature - **bandwidth-limit** setting will apply to bandwidth actually reserved for tunnel. In order to successfully cobine both features, actual bandwidth must be allowed to fluctuate to some extent - e.g. if **bandwidth-limit** will be configured to 100% (this effectively means that rate will be limited to the bandwidth reserved for tunnel), tunnel will not have any chance to increase its reservation. Therefore either **bandwidth-limit** should be configured to more than 100%, or **auto-bandwidth-reserve** should be configured to more than 0%.

Manual:MPLS/Traffic-eng

Applies to RouterOS: v3, v4 +

Interface

Sub-menu: `/mpls traffic-eng interface`

Properties:

Property	Description
`bandwidth` (*integer[bps]*; Default: **0bps**)	Total bandwidth that can be allocated on an interface by TE tunnels.
`blockade-k-factor` (*integer*; Default: **3**)	Value used to calculate blockade state timeout.
`comment` (*string*; Default:)	Short description of the item
`disabled` (*yes* | *no*; Default: **yes**)	Defines whether item is ignored or used. By default VPLS interface is disabled.
`down-flood-thresholds` (*integer[0..100],interer[0..100],...*; Default: **15,30,45,60,75,80,85,90,95,97,98,99,100**)	
`igp-flood-period` (*time*; Default: **3m**)	
`interface` (*string*; Default:)	Name of an interface on which to run RSVP.
`k-factor` (*integer*; Default: **3**)	Value used to calculate RSVP timeout. Timeout is calculated using following formula: `(K + 0.5)*1.5*R`, where K is `k-factor`, R is `refresh-time`. `Read more >>`
`refresh-time` (*time*; Default: **30s**)	Interval in which RSVP Path messages are sent out.
`resource-class` (*integer[0..FFFFFFFF]*; Default: **0**)	
`te-metric` (*integer*; Default: **1**)	

up-flood-thresholds
(*integer[0..100],interer[0..100],...*; Default: **15,30,45,60,75,80,85,90,95,97,98,99,100**)

use-udp (*yes* | *no*; Default: **no**) An RSVP implementation generally requires the ability to perform "raw" network I/O, i.e., to send and receive IP datagrams using protocol 46. Some systems may not support raw network I/O, in such cases RSVP messages can be encapsulated in UDP datagrams. Ports 1698 and 1699 will be used.

Read-only properties:

Property	Description
remaining-bw (*integer[bps]*)	Shows currently unallocated bandwidth.

Tunnel Path

Sub-menu: `/mpls traffic-eng tunnel-path`

Properties:

Property	Description
affinity-exclude (*integer*; Default:)	Do not use the path if `resource-class` matches any of specified bits.
affinity-include-all (*integer*; Default:)	Use the path only if `resource-class` matches all of specified bits.
affinity-include-any (*integer*; Default:)	Use the path if `resource-class` matches any of specified bits.
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **yes**)	Defines whether item is ignored or used. By default VPLS interface is disabled.
holding-priority (*integer[0..7]*; Default:)	Is used to decide whether this path can be preempted by another path. 0 sets the highest priority.
hops (*Address:[strict|loose] [, Address:[strinct|loose]]*; Default:)	List of hops that path traverses. Used if `use-cspf` is not enabled. It is possible to specify **strict** hop or **loose** hop: • `strict` - defines that there must not be any other hops between previous hop and "strict" hop (fully specified path). • `loose` - there are acceptable other hops between previous hop and defined hop (not fully specified path). `Read more >>`
name (*string*; Default:)	Descriptive name of tunnel path
record-route (*yes* | *no*; Default:)	If enabled, the sender node will receive information about the actual route that the LSP tunnel traverses. Record Route is analogous to a path vector, and hence can be used for loop detection.
reoptimize-interval (*time*; Default:)	Interval in which tunnel path will be re-optimized. Useful if `use-cspf` is set to **yes**.
setup-priority (*integer[0..7]*; Default:)	Parameter is used to decide whether this path can preempt another path. 0 sets the highest priority.
use-cspf (*yes* | *no*; Default: **yes**)	Whether to use CSPF to create dynamic tunnel path.

Monitoring TE Status

Path State

Sub-menu: `/mpls traffic-eng path-state`

Available read only properties:

Property	Description
bandwidth (*integer[bps]*)	Bandwidth required for the path
dst (*address:integer*)	Shows TE path destination address and tunnel ID.
egress (*yes* | *no*)	Shows if router is egress router of the path
forwarding (*yes* | *no*)	Shows if router is forwarding router of the path
in-interface (*string*)	Interface on which path message is received.
in-previous-hop (*IP*)	Recorded previous hop
label (*integer*)	
locally-originated (*yes* | *no*)	Shows if router is ingress router of the path
out-interface (*string*)	Interface through which path message is sent out.
out-label (*integer*)	
out-next-hop (*IP*)	
path-in-explicit-route ()	
path-in-record-route (*List of IPs*)	Received recorded routes along the path.
path-out-explicit-route ()	
path-out-record-route ()	List of recorded routes along the path that is sent out to next hop.
resv-bandwidth (*integer[bps]*)	bandwidth that TE path is reserving.
resv-out-record-route ()	
sending-path (*yes* | *no*)	Whether path messages are being sent
sending-resv (*yes* | *no*)	Whether resv messages are being sent
src (*Address:ID*)	Shows source address and LSP ID number

Resv State

Sub-menu: `/mpls traffic-eng resv-state`

Available read only properties:

Property	Description
active (*yes* | *no*)	Shows whether reservation is active.
bandwidth (*integer[bps]*)	Bandwidth that RSVP session is allocating.
dst (*address:ID*)	Shows TE destination address and tunnel ID from RSVP session.
egress (*yes* | *no*)	Shows if router is egress router of the path
interface (*string*)	Shows an interface on which bandwidth is reserved
label (*integer*)	
next-hop ()	
non-output (*yes* | *no*)	
recorded-route (*IP[label]*)	Shows recorded routes and labels along LSP.
shared (*yes* | *no*)	Whether LSP tunnels can share resources, so that the new LSP tunnel can be set up without having to wait for the old LSP tunnel to be cleared. Read more >>
src (*address:ID*)	Shows TE source address and LSP ID from RSVP session.

Manual:Interface/Traffic Engineering

Applies to RouterOS: v3, v4+

Properties

Sub-menu: `/interface traffic-eng`

Property	Description
affinity-exclude (*integer*; Default:)	Do not use interface if resource-class matches any of specified bits.
affinity-include-all (*integer*; Default:)	Use interface only if resource-class matches all of specified bits.
affinity-include-any (*integer*; Default:)	Use interface if resource-class matches any of specified bits.
auto-bandwidth-avg-interval (*time*; Default: **5m**)	Interval in which actual amount of data is measured, from which average bandwidth is calculated.
auto-bandwidth-range (*Disabled* | *Min[bps][-Max[bps]]*; Default: **0bps**)	Auto bandwidth adjustment range. Read more >>
auto-bandwidth-reserve (*integer[%]*; Default: **0%**)	Specifies percentage of additional bandwidth to reserve. Read more >>
auto-bandwidth-update-interval (*time*; Default: **1h**)	Interval during which tunnel keeps track of highest average rate.
bandwidth (*integer[bps]*; Default: **0bps**)	How much bandwidth to reserve for TE tunnel. Value is in bits per second. Read more >>
bandwidth-limit (*disabled* | *integer[%]*; Default: **disabled**)	Defines actual bandwidth limitation of TE tunnel. Limit is configured in percent of specified tunnel `bandwidth`. Read more >>
comment (*string*; Default:)	Short description of the item

disable-running-check (*yes* | *no*; Default: **no**)	Specifies whether to detect if interface is running or not. If set to **no** interface will always have `running` flag.
disabled (*yes* | *no*; Default: **yes**)	Defines whether item is ignored or used.
from-address (*auto* | *IP*; Default: **auto**)	Ingress address of the tunnel. If set to **auto** least IP address is picked.
holding-priority (*integer [0..7]*; Default:)	Is used to decide whether this session can be preempted by another session. 0 sets the highest priority.
mtu (*integer*; Default:)	
name (*string*; Default:)	Name of the interface
primary-path (*string*; Default:)	Primary label switching paths defined in `/mpls traffic-eng tunnel-path` menu.
primary-retry-interval (*time*; Default: **1m**)	Interval after which tunnel will try to use primary path.
record-route (*yes* | *no*; Default:)	If enabled, the sender node will receive information about the actual route that the LSP tunnel traverses. Record Route is analogous to a path vector, and hence can be used for loop detection.
reoptimize-interval (*time*; Default:)	Interval after which tunnel will re-optimize current path. If current path is not the best path then after optimization best path will be used. `Read more >>`
secondary-path (*string[,string]*; Default:)	List of label switching paths used by TE tunnel if primary path fails. Paths are defined in `/mpls traffic-eng tunnel-path` menu.
setup-priority (*integer[0..7]*; Default:)	Parameter is used to decide whether this session can preempt another session. 0 sets the highest priority.
to-address (*IP*; Default: **0.0.0.0**)	Remote end of TE tunnel.

Monitoring

To verify TE tunnel's status **monitor** command can be used.

```
[admin@R3] /interface traffic-eng> monitor 0
             tunnel-id: 12
    primary-path-state: on-hold
  secondary-path-state: established
        secondary-path: static
           active-path: static
          active-lspid: 3
          active-label: 66
        explicit-route: "S:192.168.55.10/32,L:192.168.55.13/32,L:192.168.55.17/32"
        recorded-route: "192.168.55.13[66],192.168.55.17[59],192.168.55.18[3]"
    reserved-bandwidth: 5.0Mbps
```

Reoptimization

Path can be re-optimized manually by entering following command `/interface traffic-eng reoptimize [id]`. It allows network administrators to reoptimize the LSPs that have been established based on changes in bandwidth, traffic, management policy, or other factors.

Lets say TE tunnel chose another path after link failure on best path. You can verify optimization by looking at **explicit-route** or **recorded-route** if `record-route` parameter is enabled.

```
[admin@R3] /interface traffic-eng> monitor 0
             tunnel-id: 12
    primary-path-state: established
          primary-path: dyn
  secondary-path-state: not-necessary
           active-path: dyn
          active-lspid: 1
          active-label: 67
        explicit-route: "S:192.168.55.10/32,S:192.168.55.13/32,S:192.168.55.14/32,
                         S:192.168.55.17/32,S:192.168.55.18/32"
        recorded-route: "192.168.55.13[67],192.168.55.17[60],192.168.55.18[3]"
    reserved-bandwidth: 5.0Mbps
```

Whenever the link comes back, TE tunnel will use the same path even it is not the best path (unless `reoptimize-interval` is configured). To fix it we can manually reoptimize tunnel path.

```
[admin@R3] /interface traffic-eng> reoptimize 0

[admin@R3] /interface traffic-eng> monitor 0
             tunnel-id: 12
    primary-path-state: established
          primary-path: dyn
  secondary-path-state: not-necessary
           active-path: dyn
          active-lspid: 2
          active-label: 81
        explicit-route: "S:192.168.55.5/32,S:192.168.55.2/32,S:192.168.55.1/32"
        recorded-route: "192.168.55.2[81],192.168.55.1[3]"
    reserved-bandwidth: 5.0Mbps
```

Notice how explicit-route and recorded-route changed to shorter path.

See Also

- TE Tunnel Auto Bandwidth
- TE tunnels explained

[Top | Back to Content]

Manual:Router AAA

Applies to RouterOS: 2.9, v3, v4, v5+

Summary

Sub-menu: `/user`

MikroTik RouterOS router user facility manage the users connecting the router from the local console, via serial terminal, telnet, SSH or Winbox. The users are authenticated using either local database or designated RADIUS server.

Each user is assigned to a user group, which denotes the rights of this user. A group policy is a combination of individual policy items.

In case the user authentication is performed using RADIUS, the RADIUS Client should be previously configured.

User Groups

Sub-menu: `/user group`

The router user groups provide a convenient way to assign different permissions and access rights to different user classes.

Properties

Property	Description
name (*string*; Default:)	The name of the user group
policy (*local* | *telnet* | *ssh* | *ftp* | *reboot* | *read* | *write* | *policy* | *test* | *web* | *sniff* | *api* | *winbox* | *password* | *sensitive*; Default:)	List of allowed policies:

- `local` - policy that grants rights to log in locally via console
- `telnet` - policy that grants rights to log in remotely via telnet
- `ssh` - policy that grants rights to log in remotely via secure shell protocol
- `ftp` - policy that grants full rights to log in remotely via FTP and to transfer files from and to the router. Users with this policy can both read, write and erase files, regardless of "read/write" permission, as that deals only with RouterOS configuration.
- `reboot` - policy that allows rebooting the router
- `read` - policy that grants read access to the router's configuration. All console commands that do not alter router's configuration are allowed. Doesn't affect FTP
- `write` - policy that grants write access to the router's configuration, except for user management. This policy does not allow to read the configuration, so make sure to enable read policy as well
- `policy` - policy that grants user management rights. Should be used together with write policy. Allows also to see global variables created by other users (requires also 'test' policy).
- `test` - policy that grants rights to run ping, traceroute, bandwidth-test, wireless scan, sniffer, snooper and other test commands
- `web` - policy that grants rights to log in remotely via WebBox
- `winbox` - policy that grants rights to log in remotely via WinBox
- `password` - policy that grants rights to change the password
- `sensitive` - grants rights to see sensitive information in the router, see below list as to what is regarded as sensitive.
- `api` - grants rights to access router via API.
- `sniff` - policy that grants rights to use packet sniffer tool.

Sensitive information

Starting with RouterOS v3.27, the following information is regarded as sensitive, and can be hidden from certain user groups with the **'sensitive**' policy unchecked.

Also, since RouterOS v4.3, backup files are considered sensitive, and users without this policy will not be able to download them in any way.

system package

```
/radius: secret
/snmp/community: authentication-password, encryption-password
```

advanced-tools package

```
/tool/sms: secret
```

wireless package

```
/interface/wireless/security-profiles: wpa-pre-shared-key,
wpa2-pre-shared-key, static-key-0, static-key-1, static-key-2,
static-key-3, static-sta-private-key
/interface/wireless/access-list: private-key, private-pre-shared-key
```

wireless-test package

```
/interface/wireless/security-profiles: wpa-pre-shared-key, wpa2-pre-shared-key,
static-key-0, static-key-1, static-key-2, static-key-3, static-sta-private-key, management-protection-key
/interface/wireless/access-list: private-key, private-pre-shared-key, management-protection-key
```

user-manager package

```
/tool/user-manager/user: password
/tool/user-manager/customer: password
```

hotspot package

```
/ip/hotspot/user: password
```

ppp package

```
/ppp/secret: password
```

security package

```
/ip/ipsec/installed-sa: auth-key, enc-key
/ip/ipsec/manual-sa: ah-key, esp-auth-key, esp-enc-key
/ip/ipsec/peer: secret
```

routing package

```
/routing/bgp/peer: tcp-md5-key
/routing/rip/interface: authentication-key
/routing/ospf/interface: authentication-key
/routing/ospf/virtual-link: authentication-key
```

routing-test package

```
/routing/bgp/peer: tcp-md5-key
/routing/rip/interface: authentication-key
/routing/ospf/interface: authentication-key
/routing/ospf/virtual-link: authentication-key
```

Notes

There are three system groups which cannot be deleted:

```
[admin@rb13] > /user group print
 0 name="read" policy=local,telnet,ssh,reboot,read,test,winbox,password,web,!ftp,!write,!policy

 1 name="write" policy=local,telnet,ssh,reboot,read,write,test,winbox,password,web,!ftp,!policy

 2 name="full" policy=local,telnet,ssh,ftp,reboot,read,write,policy,test,winbox,password,web

 3 name="test" policy=ssh,read,policy,!local,!telnet,!ftp,!reboot,!write,!test,!winbox,!password,!web
[admin@rb13] >
```

Exclamation sign '!' just before policy item name means NOT.

Example

To add reboot group that is allowed to reboot the router locally or using telnet, as well as read the router's configuration, enter the following command:

```
[admin@rb13] user group> add name=reboot policy=telnet,reboot,read,local
[admin@rb13] user group> print
 0 name="read" policy=local,telnet,ssh,reboot,read,test,winbox,password,web,!ftp,!write,!policy

 1 name="write" policy=local,telnet,ssh,reboot,read,write,test,winbox,password,web,!ftp,!policy
```

```
 2 name="full" policy=local,telnet,ssh,ftp,reboot,read,write,policy,test,winbox,password,web

 3 name="reboot" policy=local,telnet,reboot,read,!ssh,!ftp,!write,!policy,!test,!winbox,!password,!web
[admin@rb13] user group>
```

Router Users

Sub-menu: `/user`

Router user database stores the information such as username, password, allowed access addresses and group about router management personnel.

Properties

Property	Description
address (*IP/mask | IPv6 prefix*; Default:)	Host or network address from which the user is allowed to log in
group (*string*; Default:)	Name of the group the user belongs to
name (*string*; Default:)	User name. Although it must start with an alphanumeric character, it may contain "*", "_", "." and "@" symbols.
password (*string*; Default:)	User password. If not specified, it is left blank (hit [Enter] when logging in). It conforms to standard Unix characteristics of passwords and may contain letters, digits, "*" and "_" symbols.

Notes

There is one predefined user with full access rights:

```
[admin@MikroTik] user> print
Flags: X - disabled
  #    NAME                                              GROUP ADDRESS
  0    ;;; system default user
       admin                                             full  0.0.0.0/0

[admin@MikroTik] user>
```

There always should be at least one user with fulls access rights. If the user with full access rights is the only one, it cannot be removed.

Monitoring Active Users

Sub-menu: `/user active`

`/user active print` command shows the currently active users along with respective statisics information.

Properties

All properties are read-only.

Property	Description
address (*IP/IPv6 address*)	Host IP/IPv6 address from which the user is accessing the router. 0.0.0.0 means that user is logged in locally
group (*string*)	Group that user belongs to.
name (*string*)	User name.
radius (*true* | *false*)	Whether user is authenticated by RADIUS server.
via (*console* | *telnet* | *ssh* |*winbox* | *api* | *web*)	User's access method
when (*time*)	Time and date when user logged in.

Example

To print currently active users, enter the following command:

```
[admin@dzeltenais_burkaans] /user active> print detail
Flags: R - radius
 0   when=dec/08/2010 16:19:24 name="admin" address=10.5.8.52 via=winbox

 2   when=dec/09/2010 09:23:04 name="admin" address=10.5.101.38 via=telnet

 3   when=dec/09/2010 09:34:27 name="admin" address=fe80::21a:4dff:fe5d:8e56 via=api
```

Remote AAA

Sub-menu: `/user aaa`

Router user remote AAA enables router user authentication and accounting via RADIUS server. The RADIUS user database is consulted only if the required username is not found in the local user database

Properties

Property	Description
accounting (*yes* | *no*; Default: **yes**)	
exclude-groups (*list of group names*; Default:)	Exclude-groups consists of the groups that should not be allowed to be used for users authenticated by radius. If radius server provides group specified in this list, default-group will be used instead. This is to protect against privilege escalation when one user (without policy permission) can change radius server list, setup it's own radius server and log in as admin.
default-group (*string*; Default: **read**)	User group used by default for users authenticated via RADIUS server.
interim-update (*time*; Default: **0s**)	Interim-Update time interval
use-radius (*yes* |*no*; Default: **no**)	Enable user authentication via RADIUS

Note: If you are using RADIUS, you need to have CHAP support enabled in the RADIUS server for Winbox to work

SSH Keys

Sub-menu: `/user ssh-keys`

This menu allows to import public keys used for ssh authentication.

Warning: User is not allowed to login via ssh by password if ssh-keys for the user is added

Properties:

Property	Description
user (*string*; Default:)	username to which ssh key is assigned.

Read-only properties:

Property	Description
key-owner (*string*)	

When importing ssh key by `/user ssh-keys import` command you will be asked for two parameters:

- **public-key-file** - file name in routers root directory containing the key.
- **user** - name of the user to which key will be assigned

Private keys

Sub-menu: `/user ssh-keys private`

This menu is used to import and list imported private keys. Private keys are used to authenticate remote login attempts using certificates.

Read-only properties:

Property	Description
user (*string*)	
key-owner (*string*)	

When importing ssh keys from this sub menu using `/user ssh-keys private` import command you will be asked for three parameters:

- **private-key-file** - file name in routers root directory containing private key.
- **public-key-file** - file name in routers root directory containing public key.
- **user** - name of the user to which key will be assigned

Example

Read full example >>

Manual:PPP AAA

Applies to RouterOS: 2.9, v3, v4, v5

Summary

Sub-menu: `/ppp`

The MikroTik RouterOS provides scalable Authentication, Athorization and Accounting (AAA) functionality.

Local authentication is performed using the User Database and the Profile Database. The actual configuration for the given user is composed using respective user record from the User Database, associated item from the Profile Database and the item in the Profile database which is set as default for a given service the user is authenticating to. Default profile settings from the Profile database have lowest priority while the user access record settings from the User Database have highest priority with the only exception being particular IP addresses take precedence over IP pools in the local-address and remote-address settings, which described later on.

Support for RADIUS authentication gives the ISP or network administrator the ability to manage PPP user access and accounting from one server throughout a large network. The MikroTik RouterOS has a RADIUS client which can authenticate for PPP, PPPoE, PPTP, L2TP and ISDN connections. The attributes received from RADIUS server override the ones set in the default profile, but if some parameters are not received they are taken from the respective default profile.

User Profiles

Sub-menu: `/ppp profile`

PPP profiles are used to define default values for user access records stored under `/ppp secret` submenu. Settings in `/ppp secret` User Database override corresponding `/ppp profile` settings except that single IP addresses always take precedence over IP pools when specified as local-address or remote-address parameters.

Properties

Property	Description
address-list (*string*; Default:)	Address list name to which ppp assigned address will be added.
bridge (*string*; Default:)	Name of the bridge interface to which ppp interface will be added as slave port.
change-tcp-mss (*yes* | *no* | *default*; Default: **default**)	Modifies connection MSS settings: • `yes` - adjust connection MSS value • `no` - do not atjust connection MSS value • `default` - derive this value from the interface default profile; same as no if this is the interface default profile
comment (*string*; Default:)	
dhcpv6-pd-pool (*string*; Default:)	Name of the IPv6 pool which will be used by dynamically created DHCPv6-PD server when client connects. `Read more >>`
dns-server (*IP*; Default:)	IP address of the DNS server that is supplied to ppp clients
idle-timeout (*time*; Default:)	Specifies the amount of time after which the link will be terminated if there are no activity present. Timeout is not set by default
incoming-filter (*string*; Default:)	Firewall chain name for incoming packets. Specified chain gets control for each packet coming from the client. The ppp chain should be manually added and rules with action=jump jump-target=ppp should be added to other relevant chains in order for this feature to work. For more information look at the examples section
local-address (*IP address* | *pool*; Default:)	Tunnel address or name of the pool from which address is assigned to ppp interface locally.
name (*string*; Default:)	PPP profile name
only-one (*yes* | *no* | *default*; Default: **default**)	Defines whether a user is allowed to have more than one connection at a time • `yes` - a user is not allowed to have more than one connection at a time • `no` - the user is allowed to have more than one connection at a time • `default` - derive this value from the interface default profile; same as no if this is the interface default profile
outgoing-filter (*string*; Default:)	Firewall chain name for outgoing packets. Specified chain gets control for each packet going to the client. The ppp chain should be manually added and rules with action=jump jump-target=ppp should be added to other relevant chains in order for this feature to work. For more information look at the Examples section.
rate-limit (*string*; Default:)	Rate limitation in form of **rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time] [priority] [rx-rate-min[/tx-rate-min]]]]** from the point of view of the router (so "rx" is client upload, and "tx" is client download). All rates are measured in bits per second, unless followed by optional 'k' suffix (kilobits per second) or 'M' suffix (megabits per second). If tx-rate is not specified, rx-rate serves as tx-rate too. The same applies for tx-burst-rate, tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate are used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. Priority takes values 1..8, where 1 implies the highest priority, but 8 - the lowest. If rx-rate-min and tx-rate-min are not specified rx-rate and tx-rate values are used. The rx-rate-min and tx-rate-min values can not exceed rx-rate and tx-rate values.
remote-address (*IP*; Default:)	Tunnel address or name of the pool from which address is assigned to remote ppp interface.
remote-ipv6-prefix-pool (*string* | *none*; Default: **none**)	Assign prefix from IPv6 pool to the client and install corresponding IPv6 route.
session-timeout (*time*; Default:)	Maximum time the connection can stay up. By default no time limit is set.

use-compression (*yes* | *no* | *default*; Default: **default**)
Specifies whether to use data compression or not.
- `yes` - enable data compression
- `no` - disable data compression
- `default` - derive this value from the interface default profile; same as no if this is the interface default profile

use-encryption (*yes* | *no* | *default* | *require*; Default: **default**)
Specifies whether to use data encryption or not.
- `yes` - enable data encryption
- `no` - disable data encryption
- `default` - derive this value from the interface default profile; same as no if this is the interface default profile
- `require` - explicitly requires encryption

use-ipv6 (*yes* | *no* | *default* | *require*; Default: **default**)
Specifies whether to allow IPv6. By default is enabled if IPv6 package is installed.
- `yes` - enable IPv6 support
- `no` - disable IPv6 support
- `default` - derive this value from the interface default profile; same as **no** if this is the interface default profile
- `require` - explicitly requires IPv6 support

use-mpls (*yes* | *no* | *default* | *require*; Default: **default**)
Specifies whether to allow MPLS over PPP.
- `yes` - enable MPLS support
- `no` - disable MPLS support
- `default` - derive this value from the interface default profile; same as **no** if this is the interface default profile
- `require` - explicitly requires MPLS support

use-vj-compression (*yes* | *no* | *default*; Default: **default**)
Specifies whether to use Van Jacobson header compression algorithm.
- `yes` - enable Van Jacobson header compression
- `no` - disable Van Jacobson header compression
- `default` - derive this value from the interface default profile; same as no if this is the interface default profile

wins-server (*IP address*; Default:)
IP address of the WINS server to supply to Windows clients

Notes

There are two default profiles that cannot be removed:

```
[admin@rb13] ppp profile> print
Flags: * - default
 0 * name="default" use-compression=no use-vj-compression=no use-encryption=no only-one=no
     change-tcp-mss=yes
 1 * name="default-encryption" use-compression=default use-vj-compression=default use-encryption=yes
     only-one=default change-tcp-mss=default
[admin@rb13] ppp profile>
```

Use Van Jacobson compression only if you have to because it may slow down the communications on bad or congested channels.

incoming-filter and outgoing-filter arguments add dynamic jump rules to chain ppp, where the jump-target argument will be equal to incoming-filter or outgoing-filter argument in /ppp profile. Therefore, chain ppp should be manually added before changing these arguments.

`only-one` parameter is ignored if RADIUS authentication is used.

If there are more that 10 simultaneous PPP connections planned, it is recommended to turn the `change-mss` property off, and use one general MSS changing rule in mangle table instead, to reduce CPU utilization.

User Database

Sub-menu: `/ppp secret`

PPP User Database stores PPP user access records with PPP user profile assigned to each user.

Properties

Property	Description
caller-id (*string*; Default:)	For PPTP and L2TP it is the IP address a client must connect from. For PPPoE it is the MAC address (written in CAPITAL letters) a client must connect from. For ISDN it is the caller's number (that may or may not be provided by the operator) the client may dial-in from
comment (*string*; Default:)	Short description of the user.
disabled (*yes* | *no*; Default: **no**)	Whether secret will be used.
limit-bytes-in (*integer*; Default: **0**)	Maximal amount of bytes for a session that client can upload.
limit-bytes-out (*integer*; Default: **0**)	Maximal amount of bytes for a session that client can download.
local-address (*IP address*; Default:)	IP address that will be set locally on ppp interface.
name (*string*; Default:)	Name used for authentication
password (*string*; Default:)	Password used for authentication
profile (*string*; Default: **default**)	Which user profile to use.
remote-address (*IP*; Default:)	IP address that will be assigned to remote ppp interface.
remote-ipv6-prefix (*IPv6 prefix*; Default:)	IPv6 prefix assigned to ppp client. Prefix is added to ND prefix list enabling stateless address auto-configuration on ppp interface.Available starting from v5.0.
routes (*string*; Default:)	Routes that appear on the server when the client is connected. The route format is: dst-address gateway metric (for example, 10.1.0.0/ 24 10.0.0.1 1). Several routes may be specified separated with commas. This parameter will be ignored for OpenVPN.
service (*any* | *async* | *isdn* | *l2tp* | *pppoe* | *pptp* | *ovpn* | *sstp*; Default: **any**)	Specifies the services that particular user will be able to use.

Active Users

Sub-menu: `/ppp active`

This submenu allows to monitor active (connected) users.

`/ppp active print` command will show all currently connected users.

`/ppp active print stats` command will show received/sent bytes and packets

Properties

Property	Description
address (*IP address*)	IP address the client got from the server
bytes (*integer*)	Amount of bytes transfered through tis connection. First figure represents amount of transmitted traffic from the router's point of view, while the second one shows amount of received traffic.
caller-id (*string*)	For PPTP and L2TP it is the IP address the client connected from. For PPPoE it is the MAC address the client connected from.
encoding (*string*)	Shows encryption and encoding (separated with '/' if asymmetric) being used in this connection
limit-bytes-in (*integer*)	Maximal amount of bytes the user is allowed to send to the router.
limit-bytes-out (*integer*)	Maximal amount of bytes the user is allowed to send to the client.
name (*string*)	User name supplied at authentication stage
packets (*integer/integer*)	Amount of packets transfered through tis connection. First figure represents amount of transmitted traffic from the router's point of view, while the second one shows amount of received traffic
service (*async* | *isdn* | *l2tp* | *pppoe* | *pptp* | *ovpn* | *sstp*)	Type of service the user is using.
session-id (*string*)	Shows unique client identifier.
uptime (*time*)	User's uptime

Remote AAA

Sub-menu: `/ppp aaa`

Settings in this submenu allows to set RADIUS accounting and authentication. Note that RADIUS user database is consulted only if the required username is not found in local user database.

Properties

Property	Description
accounting (*yes* | *no*; Default: **yes**)	Enable RADIUS accounting
interim-update (*time*; Default: **0s**)	Interim-Update time interval
use-radius (*yes* | *no*; Default: **no**)	Enable user authentication via RADIUS. If entry in local secret database is not found, then client will be authenticated via RADIUS.

Examples

Add new profile

To add the profile ex that assigns the router itself the 10.0.0.1 address, and the addresses from the ex pool to the clients, filtering traffic coming from clients through mypppclients chain:

```
[admin@rb13] ppp profile> add name=ex local-address=10.0.0.1 remote-address=ex incoming-filter=mypppclients
[admin@rb13] ppp profile> print
Flags: * - default
 0 * name="default" use-compression=no use-vj-compression=no use-encryption=no only-one=no
     change-tcp-mss=yes
 1   name="ex" local-address=10.0.0.1 remote-address=ex use-compression=default
```

```
     use-vj-compression=default use-encryption=default only-one=default change-tcp-mss=default
     incoming-filter=mypppclients
 2 * name="default-encryption" use-compression=default use-vj-compression=default use-encryption=yes
     only-one=default change-tcp-mss=default
[admin@rb13] ppp profile>
```

Add new user

To add the user ex with password lkjrht and profile ex available for PPTP service only, enter the following command:

```
[admin@rb13] ppp secret> add name=ex password=lkjrht service=pptp profile=ex
[admin@rb13] ppp secret> print
Flags: X - disabled
 #   NAME                SERVICE CALLER-ID         PASSWORD          PROFILE            REMOTE-ADDRESS
 0   ex                  pptp                      lkjrht            ex                 0.0.0.0
[admin@rb13] ppp secret>
```

Manual:RADIUS Client

Applies to RouterOS: 2.9, v3, v4, v5

Summary

Sub-menu: `/radius`
Standards: `RADIUS RFC 2865`

RADIUS, short for Remote Authentication Dial-In User Service, is a remote server that provides authentication and accounting facilities to various network apliances. RADIUS authentication and accounting gives the ISP or network administrator ability to manage PPP user access and accounting from one server throughout a large network. The MikroTik RouterOS has a RADIUS client which can authenticate for HotSpot, PPP, PPPoE, PPTP, L2TP and ISDN connections. The attributes received from RADIUS server override the ones set in the default profile, but if some parameters are not received they are taken from the respective default profile.

The RADIUS server database is consulted only if no matching user acces record is found in router's local database.

Traffic is accounted locally with MikroTik Traffic Flow and Cisco IP pairs and snapshot image can be gathered using Syslog utilities. If RADIUS accounting is enabled, accounting information is also sent to the RADIUS server default for that service.

Radius Client

This sub-menu allows to add/remove radius clients.

Note: The order of added items in this list is significant.

Properties

Property	Description
accounting-backup (*yes* | *no*; Default: **no**)	Whether configuration is for backup RADIUS server
accounting-port (*integer [1..65535]*; Default: **1813**)	RADIUS server port used for accounting
address (*IPv4/IPv6 address*; Default: **0.0.0.0**)	IPv4 or IPv6 address of RADIUS server.
authentication-port (*integer [1..65535]*; Default: **1812**)	RADIUS server port used for authentication.
called-id (*string*; Default:)	Value depends on Point-to-Point protocol: PPPoE - service name, PPTP - server's IP address, L2TP - server's IP address.
comment (*string*; Default:)	
disabled (*yes* | *no*; Default: **no**)	
domain (*string*; Default:)	Microsoft Windows domain of client passed to RADIUS servers that require domain validation.
realm (*string*; Default:)	Explicitly stated realm (user domain), so the users do not have to provide proper ISP domain name in user name.
secret (*string*; Default:)	Shared secret used to access the RADIUS server.
service (*ppp|login|hotspot|wireless|dhcp*; Default:)	Router services that will use this RADIUS server: • `hotspot` - HotSpot authentication service • `login` - router's local user authentication • `ppp` - Point-to-Point clients authentication • `wireless` - wireless client authentication (client's MAC address is sent as User-Name) • `dhcp` - DHCP protocol client authentication (client's MAC address is sent as User-Name)
src-address (*ipv4/ipv6 address*; Default: **0.0.0.0**)	Source IP/IPv6 address of the packets sent to RADIUS server
timeout (*time*; Default: **100ms**)	Timeout after which the request should be resend

Note: Microsoft Windows clients send their usernames in form domain\username

Note: When RADIUS server is authenticating user with CHAP, MS-CHAPv1, MS-CHAPv2, it is not using shared secret, secret is used only in authentication reply, and router is verifying it. So if you have wrong shared secret, RADIUS server will accept request, but router won't accept reply. You can see that with /radius monitor command, "bad-replies" number should increase whenever somebody tries to connect.

Example

To set a RADIUS server for HotSpot and PPP services that has 10.0.0.3 IP address and ex shared secret, you need to do the following:

```
[admin@MikroTik] radius> add service=hotspot,ppp address=10.0.0.3 secret=ex
[admin@MikroTik] radius> print
Flags: X - disabled
  #    SERVICE         CALLED-ID     DOMAIN        ADDRESS         SECRET
  0    ppp,hotspot                                 10.0.0.3        ex
[admin@MikroTik] radius>
AAA for the respective services should be enabled too:
[admin@MikroTik] radius> /ppp aaa set use-radius=yes
[admin@MikroTik] radius> /ip hotspot profile set default use-radius=yes
To view some statistics for a client:
[admin@MikroTik] radius> monitor 0
             pending: 0
            requests: 10
             accepts: 4
             rejects: 1
             resends: 15
            timeouts: 5
         bad-replies: 0
    last-request-rtt: 0s
[admin@MikroTik] radius>
```

Connection Terminating from RADIUS

Sub-menu: `/radius incoming`

This facility supports unsolicited messages sent from RADIUS server. Unsolicited messages extend RADIUS protocol commands, that allow to terminate a session which has already been connected from RADIUS server. For this purpose DM (Disconnect-Messages) are used. Disconnect messages cause a user session to be terminated immediately.

Note: RouterOS doesn't support POD (Packet of Disconnect) the other RADIUS access request packet that performs a similar function as Disconnect Messages

Properties

Property	Description
accept (*yes* | *no*; Default: **no**)	Whether to accept the unsolicited messages
port (*integer*; Default: **1700**)	The port number to listen for the requests on

Supported RADIUS Attributes

Here you can download the RADIUS reference dictionary, which incorporates all the needed RADIUS attributes. This dictionary is the minimal dictionary, which is enough to support all features of MikroTik RouterOS. It is designed for FreeRADIUS [1], but may also be used with many other UNIX RADIUS servers (eg. XTRadius [2]).

Note: it may conflict with the default configuration files of RADIUS server, which have references to the Attributes, absent in this dictionary. Please correct the configuration files, not the dictionary, as no other Attributes are supported by MikroTik RouterOS.

There is also the RADIUS MikroTik specific dictionary that can be included in an existing dictionary to support MikroTik vendor-specific Attributes.

Definitions

- **PPPs** - PPP, PPTP, PPPoE and ISDN
- **default configuration** - settings in default profile (for PPPs) or HotSpot server settings (for HotSpot)

Access-Request

- **Service-Type** - always is "Framed" (only for PPPs)
- **Framed-Protocol** - always is "PPP" (only for PPPs)
- **NAS-Identifier** - router identity
- **NAS-IP-Address** - IP address of the router itself
- **NAS-Port** - unique session ID
- **Acct-Session-Id** - unique session ID
- **NAS-Port-Type** - async PPP - "Async"; PPTP and L2TP - "Virtual"; PPPoE - "Ethernet"; ISDN - "ISDN Sync"; HotSpot - "Ethernet | Cable | Wireless-802.11" (according to the value of nas-port-type parameter in /ip hotspot p
- **Calling-Station-Id** - PPPoE and HotSpot- client MAC address in capital letters; PPTP and L2TP - client public IP address; ISDN - client MSN
- **Called-Station-Id** - PPPoE - service name; PPTP and L2TP - server IP address; ISDN - interface MSN; HotSpot - name of the HotSpot server
- **NAS-Port-Id** - async PPP - serial port name; PPPoE - ethernet interface name on which server is running; HotSpot - name of the physical HotSpot interface (if bridged, the bridge port name is showed here); not present for ISDN, PPTP and L2TP
- **Framed-IP-Address** - IP address of HotSpot client after Universal Client translation
- **Mikrotik-Host-IP** - IP address of HotSpot client before Universal Client translation (the original IP address of the client)
- **User-Name** - client login name
- **MS-CHAP-Domain** - User domain, if present
- **Mikrotik-Realm** - If it is set in /radius menu, it is included in every RADIUS request as Mikrotik-Realm attribute. If it is not set, the same value is sent as in MS-CHAP-Domain attribute (if MS-CHAP-Domain is missing, Realm is not included neither)

- **WISPr-Location-ID** - text string specified in radius-location-id property of the HotSpot server
- **WISPr-Location-Name** - text string specified in radius-location-name property of the HotSpot server
- **WISPr-Logoff-URL** - full link to the login page (for example, http://10.48.0.1/lv/logout)

Depending on authentication methods (NOTE: HotSpot uses CHAP by default and may use also PAP if unencrypted passwords are enabled, it can not use MSCHAP):

- **User-Password** - encrypted password (used with PAP authentication)
- **CHAP-Password, CHAP-Challenge** - encrypted password and challenge (used with CHAP authentication)
- **MS-CHAP-Response, MS-CHAP-Challenge** - encrypted password and challenge (used with MS-CHAPv1 authentication)
- **MS-CHAP2-Response, MS-CHAP-Challenge** - encrypted password and challenge (used with MS-CHAPv2 authentication)

Access-Accept

- **Framed-IP-Address** - IP address given to client. If address belongs to 127.0.0.0/8 or 224.0.0.0/3 networks, IP pool is used from the default profile to allocate client IP address. If Framed-IP-Address is specified, Framed-Pool is ignored
- **Framed-IP-Netmask** - client netmask. PPPs - if specified, a route will be created to the network Framed-IP-Address belongs to via the Framed-IP-Address gateway; HotSpot - ignored by HotSpot
- **Framed-Pool** - IP pool name (on the router) from which to get IP address for the client. If Framed-IP-Address is specified, this attribute is ignored
- **Framed-IPv6-Prefix** - Ipv6 prefix assigned for the client. Added in v5.8
- **Mikrotik-Delegated-IPv6-Pool** - IPv6 pool used for Prefix Delegation. Added in v5.9

NOTE: if Framed-IP-Address or Framed-Pool is specified it overrides remote-address in default configuration

- **Idle-Timeout** - overrides idle-timeout in the default configuration
- **Session-Timeout** - overrides session-timeout in the default configuration
- **Port-Limit** - maximal mumber of simultaneous connections using the same username (overrides te shared-users property of the HotSpot user profile)
- **Class** - cookie, will be included in Accounting-Request unchanged
- **Framed-Route** - routes to add on the server. Format is specified in RFC 2865 (Ch. 5.22), can be specified as many times as needed
- **Filter-Id** - firewall filter chain name. It is used to make a dynamic firewall rule. Firewall chain name can have suffix .in or .out, that will install rule only for incoming or outgoing traffic. Multiple Filter-id can be provided, but only last ones for incoming and outgoing is used. For PPPs - filter rules in ppp chain that will jump to the specified chain, if a packet has come to/from the client (that means that you should first create a ppp chain and make jump rules that would put actual traffic to this chain). The same applies for HotSpot, but the rules will be created in hotspot chain
- **Mikrotik-Mark-Id** - firewall mangle chain name (HotSpot only). The MikroTik RADIUS client upon receiving this attribute creates a dynamic firewall mangle rule with action=jump chain=hotspot and jump-target equal to the atribute value. Mangle chain name can have suffixes .in or .out, that will install rule only for incoming or outgoing traffic. Multiple Mark-id attributes can be provided, but only last ones for incoming and outgoing is used.
- **Acct-Interim-Interval** - interim-update for RADIUS client. PPP - if 0 uses the one specified in RADIUS client; HotSpot - only respected if radius-interim-update=received in HotSpot server profile
- **MS-MPPE-Encryption-Policy** - require-encryption property (PPPs only)
- **MS-MPPE-Encryption-Types** - use-encryption property, non-zero value means to use encryption (PPPs only)

- **Ascend-Data-Rate** - tx/rx data rate limitation if multiple attributes are provided, first limits tx data rate, second - rx data rate. If used together with Ascend-Xmit-Rate, specifies rx rate. 0 if unlimited. Ignored if Rate-Limit attribute is present
- **Ascend-Xmit-Rate** - tx data rate limitation. It may be used to specify tx limit only instead of sending two sequental Ascend-Data-Rate attributes (in that case Ascend-Data-Rate will specify the receive rate). 0 if unlimited. Ignored if Rate-Limit attribute is present
- **MS-CHAP2-Success** - auth. response if MS-CHAPv2 was used (for PPPs only)
- **MS-MPPE-Send-Key, MS-MPPE-Recv-Key** - encryption keys for encrypted PPPs provided by RADIUS server only is MS-CHAPv2 was used as authentication (for PPPs only)
- **Ascend-Client-Gateway** - client gateway for DHCP-pool HotSpot login method (HotSpot only)
- **Mikrotik-Recv-Limit** - total receive limit in bytes for the client
- **Mikrotik-Recv-Limit-Gigawords** - 4G (2^32) bytes of total receive limit (bits 32..63, when bits 0..31 are delivered in Mikrotik-Recv-Limit)
- **Mikrotik-Xmit-Limit** - total transmit limit in bytes for the client
- **Mikrotik-Xmit-Limit-Gigawords** - 4G (2^32) bytes of total transmit limit (bits 32..63, when bits 0..31 are delivered in Mikrotik-Recv-Limit)
- **Mikrotik-Wireless-Forward** - not forward the client's frames back to the wireless infrastructure if this attribute is set to "0" (Wireless only)
- **Mikrotik-Wireless-Skip-Dot1x** - disable 802.1x authentication for the particulat wireless client if set to non-zero value (Wireless only)
- **Mikrotik-Wireless-Enc-Algo** - WEP encryption algorithm: 0 - no encryption, 1 - 40-bit WEP, 2 - 104-bit WEP (Wireless only)
- **Mikrotik-Wireless-Enc-Key** - WEP encruption key for the client (Wireless only)
- **Mikrotik-Rate-Limit** - Datarate limitation for clients. Format is: rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time] [priority] [rx-rate-min[/tx-rate-min]]]] from the point of view of the router (so "rx" is client upload, and "tx" is client download). All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate is used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. Priority takes values 1..8, where 1 implies the highest priority, but 8 - the lowest. If rx-rate-min and tx-rate-min are not specified rx-rate and tx-rate values are used. The rx-rate-min and tx-rate-min values can not exceed rx-rate and tx-rate values.
- **Mikrotik-Group** - Router local user group name (defines in /user group) for local users. HotSpot default profile for HotSpot users.
- **Mikrotik-Advertise-URL** - URL of the page with advertisements that should be displayed to clients. If this attribute is specified, advertisements are enabled automatically, including transparent proxy, even if they were explicitly disabled in the corresponding user profile. Multiple attribute instances may be send by RADIUS server to specify additional URLs which are choosen in round robin fashion.
- **Mikrotik-Advertise-Interval** - Time interval between two adjacent advertisements. Multiple attribute instances may be send by RADIUS server to specify additional intervals. All interval values are threated as a list and are taken one-by-one for each successful advertisement. If end of list is reached, the last value is continued to be used.
- **WISPr-Redirection-URL** - URL, which the clients will be redirected to after successfull login
- **WISPr-Bandwidth-Min-Up** - minimal datarate (CIR) provided for the client upload
- **WISPr-Bandwidth-Min-Down** - minimal datarate (CIR) provided for the client download
- **WISPr-Bandwidth-Max-Up** - maxmal datarate (MIR) provided for the client upload

- **WISPr-Bandwidth-Max-Down** - maxmal datarate (MIR) provided for the client download
- **WISPr-Session-Terminate-Time** - time, when the user should be disconnected; in "YYYY-MM-DDThh:mm:ssTZD" form, where Y - year; M - month; D - day; T - separator symbol (must be written between date and time); h - hour (in 24 hour format); m - minute; s - second; TZD - time zone in one of these forms: "+hh:mm", "+hhmm", "-hh:mm", "-hhmm"

Note: the received attributes override the default ones (set in the default profile), but if an attribute is not received from RADIUS server, the default one is to be used.

Rate-Limit takes precedence over all other ways to specify data rate for the client. Ascend data rate attributes are considered second; and WISPr attributes takes the last precedence.

Here are some Rate-Limit examples:

- **128k** - rx-rate=128000, tx-rate=128000 (no bursts)
- **64k/128M** - rx-rate=64000, tx-rate=128000000
- **64k 256k** - rx/tx-rate=64000, rx/tx-burst-rate=256000, rx/tx-burst-threshold=64000, rx/tx-burst-time=1s
- **64k/64k 256k/256k 128k/128k 10/10** - rx/tx-rate=64000, rx/tx-burst-rate=256000, rx/tx-burst-threshold=128000, rx/tx-burst-time=10s

Accounting-Request

The accounting request carries the same attributes as Access Request, plus these ones:

- **Acct-Status-Type** - Start, Stop, or Interim-Update
- **Acct-Authentic** - either authenticated by the RADIUS or Local authority (PPPs only)
- **Class** - RADIUS server cookie, as received in Access-Accept
- **Acct-Delay-Time** - how long does the router try to send this Accounting-Request packet

Stop and Interim-Update Accounting-Request

Additionally to the accounting start request, the following messages will contain the following attributes:

- **Acct-Session-Time** - connection uptime in seconds
- **Acct-Input-Octets** - bytes received from the client
- **Acct-Input-Gigawords** - 4G (2^32) bytes received from the client (bits 32..63, when bits 0..31 are delivered in Acct-Input-Octets)
- **Acct-Input-Packets** - nubmer of packets received from the client
- **Acct-Output-Octets** - bytes sent to the client
- **Acct-Output-Gigawords** - 4G (2^32) bytes sent to the client (bits 32..63, when bits 0..31 are delivered in Acct-Output-Octets)
- **Acct-Output-Packets** - number of packets sent to the client

Stop Accounting-Request

These packets will, additionally to the Interim Update packets, have:

- **`Acct-Terminate-Cause`** - session termination cause (see RFC 2866 ch. 5.10)

Change of Authorization

RADIUS disconnect and Change of Authorization (according to RFC3576) are supported as well. These attributes may be changed by a CoA request from the RADIUS server:

- Mikrotik-Group
- Mikrotik-Recv-Limit
- Mikrotik-Xmit-Limit
- Mikrotik-Rate-Limit
- Ascend-Data-Rate (only if Mikrotik-Rate-Limit is not present)
- Ascend-XMit-Rate (only if Mikrotik-Rate-Limit is not present)
- Mikrotik-Mark-Id
- Filter-Id
- Mikrotik-Advertise-Url
- Mikrotik-Advertise-Interval
- Session-Timeout
- Idle-Timeout
- Port-Limit

Note that it is not possible to change IP address, pool or routes that way - for such changes a user must be disconnected first.

MikroTik Specific RADIUS Attribute Numeric Values

Click here to get plain text attribute list of MikroTik specific attributes (FreeRadius comaptible) .

Name	VendorID	Value	RFC
MIKROTIK_RECV_LIMIT	14988	1	
MIKROTIK_XMIT_LIMIT	14988	2	
MIKROTIK_GROUP	14988	3	
MIKROTIK_WIRELESS_FORWARD	14988	4	
MIKROTIK_WIRELESS_SKIPDOT1X	14988	5	
MIKROTIK_WIRELESS_ENCALGO	14988	6	
MIKROTIK_WIRELESS_ENCKEY	14988	7	
MIKROTIK_RATE_LIMIT	14988	8	
MIKROTIK_REALM	14988	9	
MIKROTIK_HOST_IP	14988	10	
MIKROTIK_MARK_ID	14988	11	
MIKROTIK_ADVERTISE_URL	14988	12	
MIKROTIK_ADVERTISE_INTERVAL	14988	13	
MIKROTIK_RECV_LIMIT_GIGAWORDS	14988	14	
MIKROTIK_XMIT_LIMIT_GIGAWORDS	14988	15	
MIKROTIK_WIRELESS_PSK	14988	16	

MIKROTIK_TOTAL_LIMIT	14988	17
MIKROTIK_TOTAL_LIMIT_GIGAWORDS	14988	18
MIKROTIK_ADDRESS_LIST	14988	19
MIKROTIK_WIRELESS_MPKEY	14988	20
MIKROTIK_WIRELESS_COMMENT	14988	21
MIKROTIK_DELEGATED_IPV6_POOL	14988	22

All Supported Attribute Numeric Values

Note: FreeRadius already has these attributes predefined. If you are using other radius server then use table below to create dictionary file

Name	VendorID	Value	RFC
Acct-Authentic		45	RFC 2866
Acct-Delay-Time		41	RFC 2866
Acct-Input-Gigawords		52	RFC 2869
Acct-Input-Octets		42	RFC 2866
Acct-Input-Packets		47	RFC 2866
Acct-Interim-Interval		85	RFC 2869
Acct-Output-Gigawords		53	RFC 2869
Acct-Output-Octets		43	RFC 2866
Acct-Output-Packets		48	RFC 2866
Acct-Session-Id		44	RFC 2866
Acct-Session-Time		46	RFC 2866
Acct-Status-Type		40	RFC 2866
Acct-Terminate-Cause		49	RFC 2866
Ascend-Client-Gateway	529	132	
Ascend-Data-Rate	529	197	
Ascend-Xmit-Rate	529	255	
Called-Station-Id		30	RFC 2865
Calling-Station-Id		31	RFC 2865
CHAP-Challenge		60	RFC 2866
CHAP-Password		3	RFC 2865
Class		25	RFC 2865
Filter-Id		11	RFC 2865
Framed-IP-Address		8	RFC 2865
Framed-IP-Netmask		9	RFC 2865
Framed-IPv6-Prefix		97	RFC 3162

Framed-Pool		88	RFC 2869
Framed-Protocol		7	RFC 2865
Framed-Route		22	RFC 2865
Idle-Timeout		28	RFC 2865
MS-CHAP-Challenge	311	11	RFC 2548
MS-CHAP-Domain	311	10	RFC 2548
MS-CHAP-Response	311	1	RFC 2548
MS-CHAP2-Response	311	25	RFC 2548
MS-CHAP2-Success	311	26	RFC 2548
MS-MPPE-Encryption-Policy	311	7	RFC 2548
MS-MPPE-Encryption-Types	311	8	RFC 2548
MS-MPPE-Recv-Key	311	17	RFC 2548
MS-MPPE-Send-Key	311	16	RFC 2548
NAS-Identifier		32	RFC 2865
NAS-Port		5	RFC 2865
NAS-IP-Address		4	RFC 2865
NAS-Port-Id		87	RFC 2869
NAS-Port-Type		61	RFC 2865
Port-Limit		62	RFC 2865
Redback-Agent-Remote-Id	2352	96	
Redback-Agent-Circuit-Id	2352	97	
Service-Type		6	RFC 2865
Session-Timeout		27	RFC 2865
User-Name		1	RFC 2865
User-Password		2	RFC 2865
WISPr-Bandwidth-Max-Down	14122	8	wi-fi.org
WISPr-Bandwidth-Max-Up	14122	7	wi-fi.org
WISPr-Bandwidth-Min-Down	14122	6	wi-fi.org
WISPr-Bandwidth-Min-Up	14122	5	wi-fi.org
WISPr-Location-Id	14122	1	wi-fi.org
WISPr-Location-Name	14122	2	wi-fi.org
WISPr-Logoff-URL	14122	3	wi-fi.org
WISPr-Redirection-URL	14122	4	wi-fi.org
WISPr-Session-Terminate-Time	14122	9	wi-fi.org

Troubleshooting

My radius server accepts authentication request from the client with "Auth: Login OK:...", but the user cannot log on. The bad replies counter is incrementing under radius monitor.

> This situation can occur, if the radius client and server have high delay link between them. Try to increase the radius client's timeout to 600ms or more instead of the default 300ms! Also, double check, if the secrets match on client and server!

[Top | Back to Content]

References

[1] http://freeradius.org
[2] http://xtradius.sourceforge.net/

Manual:User Manager

Introduction

- What is User Manager
- Requirements
- Supported browsers
- Demo
- Differences between version 3 and version 4-test

Getting started

- Download
- Install
- Create first subscriber
- First log on User Manager web

Quick start

- User Manager and HotSpot
- User Manager and PPP servers
- User Manager and DHCP
- User Manager and Wireless
- User Manager and RouterOS user

Concepts explained

Common

- Customers
- Users
- Routers
- Sessions
- Payments
- Reports
- Logs
- Customer permission levels
- Character constants
- Active sessions
- Active users
- Customer public ID

Version 4.x test package specific

- Profiles
- Limitations
- User data templates
- MAC binding
- Languages
- CoA (Radius incoming)

Version 3.x specific

- Subscribers
- Credits
- User prefix
- Time, traffic amount and rate limiting
- Prepaid and unlimited users
- Voucher template

Reference

Web interface

- Search patterns
- Tables:
  - Sorting
  - Filtering
  - Division in pages
  - Multiple object selection
  - Operations with selected objects
  - Minimization
  - Links to detail form
- Detail forms
- Page printing

Customer page

- Setup
- How to find it?
- Sections
  - Status
  - Routers
  - Credits
  - Users
  - Sessions
  - Customers
  - Reports
  - Logs

User page

- Setup
- How to find it?
- Link to user page
- Sections
  - Status
  - Payments
  - Settings

User sign-up

- Setup
- Sign-up steps
  - Creating account
  - Activating account
  - Login

User payments

- Authorize.Net
- PayPal

Questions and answers

- Quick introduction into User Manager setup
- How to separate users among customers?
- How to create a link to user page?
- How to create a link to user sign-up page?
- Visual bugs since upgrade
- Cannot log in User Manager
- Too many active sessions shown
- What does "active sessions" refer to?
- How to make Hotspot and User Manager on the same router?
- How to make MAC authentication in the User Manager?
- How to turn off logging for specific Routers?

- How to create timed Voucher?
- Cannot access User Manager WEB interface
- Incorrect time shown for sessions and credits
- User Manager does not allow to login due to expired uptime
- How to debug PayPal payments
- How to send logs to a remote host, using SysLog

Manual:Hotspot Introduction

Summary

HotSpot is a way to authorize users to access some network resources, but does not provide traffic encryption. To log in, users may use almost any web browser (either HTTP or HTTPS protocol), so they are not required to install additional software. The gateway is accounting the uptime and amount of traffic each client have used, and also can send this information to a RADIUS server. The HotSpot system may limit each particular user's bitrate, total amount of traffic, uptime and some other parameters mentioned further in this document.

The HotSpot system is targeted to provide authentication within a local network (for the local network users to access the Internet), but may as well be used to authorize access from outer networks to access local resources (like an authentication gateway for the outside world to access your network). It is possible to allow users to access some web pages without authentication using Walled Garden feature.

Getting an Address

First of all, a client have to get an IP address. It may be set on the client statically, or leased from a DHCP server. The DHCP server may provide ways of binding lent IP addresses to clients MAC addresses, if required. The HotSpot system does not care how client get an address before he/she gets to the HotSpot login page.

Moreover, HotSpot server may automatically and transparently change any IP address (yes, meaning really any IP address) of a client to a valid unused address from the selected IP pool. If a user is able to get his/her Internet connection working at their place, he/she will be able to get his/her connection working in the HotSpot network. This feature gives a possibility to provide a network access (for example, Internet access) to mobile clients that are not willing (or are disallowed, not qualified enough or otherwise unable) to change their networking settings. The users will not notice the translation (i.e., there will not be any changes in the users' config), but the router itself will see completely different (from what is actually set on each client) source IP addresses on packets sent from the clients (even the firewall mangle table will 'see' the translated addresses). This technique is called **one-to-one NAT**, but is also known as "Universal Client" as that is how it was called in the RouterOS version 2.8.

One-to-one NAT accepts any incoming address from a connected network interface and performs a network address translation so that data may be routed through standard IP networks. Clients may use any preconfigured addresses. If the one-to-one NAT feature is set to translate a client's address to a public IP address, then the client may even run a server or any other service that requires a public IP address. This NAT is changing source address of each packet just after it is received by the router (it is like source NAT that is performed early in the packet path, so that even firewall mangle table, which normally 'sees' received packets unaltered, can only 'see' the translated address).

Note: arp mode must be enabled on the interface where one-to-one NAT is used

Before the authentication

When enabling HotSpot on an interface, the system automatically sets up everything needed to show login page for all clients that are not logged in. This is done by adding dynamic destination NAT rules, which you can observe on a working HotSpot system. These rules are needed to redirect all HTTP and HTTPS requests from unauthorized users to the HotSpot authentication proxy. Other rules that are also inserted, will be described later in a special section of this manual.

In most common setup, opening any HTTP page will bring up the HotSpot servlet login page (which can be customized extensively, as described later on). As normal user behavior is to open web pages by their DNS names, a valid DNS configuration should be set up on the HotSpot gateway itself (it is possible to reconfigure the gateway so that it will not require local DNS configuration, but such a configuration is impractical and thus not recommended).

Walled Garden

You may wish not to require authorization for some services (for example to let clients access the web server of your company without registration), or even to require authorization only to a number of services (for example, for users to be allowed to access an internal file server or another restricted area). This can be done by setting up Walled Garden system.

When a not logged-in user requests a service allowed in the Walled Garden configuration, the HotSpot gateway does not intercept it, or in case of HTTP, simply redirects the request to the original destination. Other requests are redirected to the HotSpot servlet (login page infrastructure). When a user is logged in, there is no effect of this table on him/her.

Walled Garden for HTTP requests is using the embedded proxy server . This means that all the configured parameters of that proy server will also be effective for the WalledGarden clients (as well as for all clients that have transparent proxy enabled)

Authentication

There are currently 6 different authentication methods. You can use one or more of them simultaneously:

- **HTTP PAP** - simplest method, which shows the HotSpot login page and expect to get the authentication info (i.e. username and password) in plain text. Another use of this method is the possibility of hard-coded authentication information in the servlet's login page simply creating the appropriate link.

Note: passwords are not encrypted when transferred over the network

- **HTTP CHAP** - standard method, which includes CHAP challenge in the login page. The CHAP MD5 hash challenge is used together with the user's password for computing the string which will be sent to the HotSpot gateway. The hash result (as a password) together with username is sent over network to HotSpot service (so, password is never sent in plain text over IP network). On the client side, MD5 algorithm is implemented in JavaScript applet, so if a browser does not support JavaScript (like, for example, Internet Explorer 2.0 or some PDA browsers) or it has JavaScipt disabled, it will not be able to authenticate users. It is possible to allow unencrypted passwords to be accepted by turning on HTTP PAP authentication method, but it is not recommended due to security considerations.
- **HTTPS** - the same as HTTP PAP, but uses SSL protocol to encrypt transmissions. HotSpot user just sends his/her password without additional hashing (note that there is no need to worry about plain-text password exposure over the network, as the transmission itself is encrypted). In either case, **HTTP POST** method (if not possible, then - **HTTP GET** method) is used to send data to the HotSpot gateway.
- **HTTP cookie** - after each successful login, a cookie is sent to the web browser and the same cookie is added to active HTTP cookie list. Next time the same user will try to log in, web browser will send the saved HTTP cookie. This cookie will be compared with the one stored on the HotSpot gateway and only if source MAC

address and randomly generated ID matches the ones stored on the gateway, user will be automatically logged in using the login information (username and password pair) was used when the cookie was first generated. Otherwise, the user will be prompted to log in, and in the case authentication is successful, old cookie will be removed from the local HotSpot active cookie list and the new one with different random ID and expiration time will be added to the list and sent to the web browser. It is also possible to erase cookie on user manual logoff (not in the default server pages, but you can modify them to perform this). This method may only be used together with HTTP PAP, HTTP CHAP or HTTPS methods as there would be nothing to generate cookies in the first place otherwise.

- **MAC address** - try to authenticate clients as soon as they appear in the hosts list (i.e., as soon as they have sent any packet to the HotSpot server), using client's MAC address as username.
- **Trial** - users may be allowed to use the service free of charge for some period of time for evaluation, and be required to authenticate only after this period is over. HotSpot can be configured to allow some amount of time per MAC address to be freely used with some limitations imposed by the provided user profile. In case the MAC address still has some trial time unused, the login page will contain the link for trial login. The time is automatically reset after the configured amount of time (so that, for example, any MAC address may use 30 minutes a day without ever registering). The username of such a user (as seen in the active user table and in the login link) is "T-XX:XX:XX:XX:XX:XX" (where XX:XX:XX:XX:XX:XX is his/her MAC address). The authentication procedure will not ask RADIUS server permission to authorise such a user.

HotSpot can authenticate users consulting the local user database or a RADIUS server (local database is consulted first, then - a RADIUS server). In case of HTTP cookie authentication via RADIUS server, the router will send the same information to the server as it was used when the cookie was first generated. If authentication is done locally, profile corresponding to that user is used, otherwise (in case RADIUS reply did not contain the group for that user) the default profile is used to set default values for parameters, which are not set in RADIUS access-accept message. For more information on how the interaction with a RADIUS server works, see the respective manual section.

The HTTP PAP method also makes it possible to authenticate by requesting the page:

`/login?username=username&password=password`

In case you want to log in using telnet connection, the exact HTTP request would look like that:

```
GET /login?username=username&password=password HTTP/1.0
```

Note that the request is case-sensitive.

Authorization

After authentication user gets access to the Internet and receives some limitations (which are user profile specific). HotSpot may also perform a one-to-one NAT for the client, so that a particular user would always receive the same IP address regardless of what PC is used.

The system will automatically detect and redirect requests to a proxy server that client is using (if any; it may be set in his/her settings to use an unknown proxy server) to the proxy server embedded in the router.

Authorization may be delegated to a RADIUS server, which delivers similar configuration options as the local database. For any user requiring authorization, a RADIUS server gets queried first, and if no reply received, the local database is examined. RADIUS server may send a Change of Authorization request according to standards to alter the previously accepted parameters.

MAC Cookie

MAC cookie is a new hotspot feature, designed to improve accessibility for smartphones, laptops and other mobile devices.

When MAC cookie feature is enabled (**`login-by`**`=mac-cookie`, **`add-mac-cookie`**`=yes` set in user profile), following actions are taken:

- **first successful login**. Mac cookie keeps record of username and password for the MAC address if there is only one host with such MAC. Cookie timeout is set to value equal to `mac-cookie-timeout`.
- **new host appears**. Hotspot checks if there is a mac cookie record for the MAC address and logs in host using recorded username and password. If there is more than one host with the same MAC address, user will not be logged in and MAC cookie record for this address will be deleted.
- When **user logs out** mac cookie is removed in following cases:
  - user-request - user clicked on logout button.
  - admin-reset - disconnected from radius server or user is removed from hotspot active menu
  - nas-request - traffic limit reached
  - session-timeout

Advertisement

The same proxy used for unauthorized clients to provide Walled-Garden facility, may also be used for authorized users to show them advertisement popups. Transparent proxy for authorized users allows to monitor http requests of the clients and to take some action if required. It enables the possibility to open status page even if client is logged in by mac address, as well as to show advertisements time after time

When the time has come to show an advertisement, the server redirects client's web browser to the status page. Only requests, which provide html content, are redirected (images and other content will not be affected). The status page displays the advertisement and next advertise-interval is used to schedule next advertisement. If status page is unable to display an advertisement for configured timeout starting from moment, when it is scheduled to be shown, client access is blocked within walled-garden (just as unauthorized clients are). Client is unblocked when the scheduled page is finally shown. Note that if popup windows are blocked in the browser, the link on the status page may be used to open the advertisement manually.

While client is blocked, FTP and other services are not allowed. Thus requiring client to open an advertisement for any Internet activity not especially allowed by the Walled-Garden.

Accounting

The HotSpot system implement accounting internally, you are not required to do anything special for it to work. The accounting information for each user may be sent to a RADIUS server.

Configuration menus

- /ip hotspot - HotSpot servers on particular interfaces (one server per interface). HotSpot server must be added in this menu in order for HotSpot system to work on an interface /ip hotspot profile - HotSpot server profiles. Settings, which affect login procedure for HotSpot clients are configured here. More than one HotSpot servers may use the same profile
- /ip hotspot host - dynamic list of active network hosts on all HotSpot interfaces. Here you can also find IP address bindings of the one-to-one NAT
- /ip hotspot ip-binding - rules for binding IP addresses to hosts on hotspot interfaces
- /ip hotspot service-port - address translation helpers for the one-to-one NAT

- /ip hotspot walled-garden - Walled Garden rules at HTTP level (DNS names, HTTP request substrings)
- /ip hotspot walled-garden ip - Walled Garden rules at IP level (IP addresses, IP protocols)
- /ip hotspot user - local HotSpot system users
- /ip hotspot user profile - local HotSpot system users profiles (user groups)
- /ip hotspot active - dynamic list of all authenticated HotSpot users
- /ip hotspot cookie - dynamic list of all valid HTTP cookies

[Top | Back to Content]

Manual:Customizing Hotspot

Applies to RouterOS: v3, v4, v5+

HTML customizations

Summary

You can create a completely different set of servlet pages for each HotSpot server you have, specifying the directory it will be stored in html-directory property of a HotSpot server profile /ip hotspot profile. The default servlet pages are copied in the directory of your choice right after you create the profile. This directory can be accessed by connecting to the router with an FTP client. You can modify the pages as you like using the information from this section of the manual. Note that it is suggested to edit the files manually, as automated HTML editing tools may corrupt the pages by removing variables or other vital parts.

Available Pages

Main HTML servlet pages, which are shown to user:

- **redirect.html** - redirects user to another url (for example, to login page)
- **login.html** - login page shown to a user to ask for username and password. This page may take the following parameters:
  - **username** - username
  - **password** - either plain-text password (in case of PAP authentication) or MD5 hash of chap-id variable, password and CHAP challenge (in case of CHAP authentication). This value is used as e-mail address for trial users
  - **dst** - original URL requested before the redirect. This will be opened on successfull login
  - **popup** - whether to pop-up a status window on successfull login
  - **radius<id>** - send the attribute identified with <id> in text string form to the RADIUS server (in case RADIUS authentication is used; lost otherwise)
  - **radius<id>u** - send the attribute identified with <id> in unsigned integer form to the RADIUS server (in case RADIUS authentication is used; lost otherwise)
  - **radius<id>-<vnd-id>** - send the attribute identified with <id> and vendor ID <vnd-id> in text string form to the RADIUS server (in case RADIUS authentication is used; lost otherwise)
  - **radius<id>-<vnd-id>u** - send the attribute identified with <id> and vendor ID <vnd-id> in unsigned integer form to the RADIUS server (in case RADIUS authentication is used; lost otherwise)
- **md5.js** - JavaScript for MD5 password hashing. Used together with http-chap login method

- **alogin.html** - page shown after client has logged in. It pops-up status page and redirects browser to originally requested page (before he/she was redirected to the HotSpot login page)
- **status.html** - status page, shows statistics for the client. It is also able to display advertisements automatically
- **logout.html** - logout page, shown after user is logged out. Shows final statistics about the finished session. This page may take the following additional parameters:
  - **erase-cookie** - whether to erase cookies from the HotSpot server on logout (makes impossible to log in with cookie next time from the same browser, might be useful in multiuser environments)
- **error.html** - error page, shown on fatal errors only

Some other pages are available as well, if more control is needed:

- **rlogin.html** - page, which redirects client from some other URL to the login page, if authorization of the client is required to access that URL
- **rstatus.html** - similarly to rlogin.html, only in case if the client is already logged in and the original URL is not known
- **radvert.html** - redirects client to the scheduled advertisement link
- **flogin.html** - shown instead of login.html, if some error has happened (invalid username or password, for example)
- **fstatus.html** - shown instead of redirect, if status page is requested, but client is not logged in
- **flogout.html** - shown instead of redirect, if logout page is requested, but client is not logged in

Serving Servlet Pages

The HotSpot servlet recognizes 5 different request types:

1. **request for a remote host**
   - if user is logged in and advertisement is due to be displayed, radvert.html is displayed. This page makes redirect to the scheduled advertisment page
   - if user is logged in and advertisement is not scheduled for this user, the requested page is served
   - if user is not logged in, but the destination host is allowed by walled garden, then the request is also served
   - if user is not logged in, and the destination host is disallowed by walled garden, rlogin.html is displayed; if rlogin.html is not found, redirect.html is used to redirect to the login page
2. **request for "/" on the HotSpot host**
   - if user is logged in, rstatus.html is displayed; if rstatus.html is not found, redirect.html is used to redirect to the status page
   - if user is not logged in, rlogin.html is displayed; if rlogin.html is not found, redirect.html is used to redirect to the login page
3. **request for "/login" page**
   - if user has successfully logged in (or is already logged in), alogin.html is displayed; if alogin.html is not found, redirect.html is used to redirect to the originally requested page or the status page (in case, original destination page was not given)
   - if user is not logged in (username was not supplied, no error message appeared), login.html is showed
   - if login procedure has failed (error message is supplied), flogin.html is displayed; if flogin.html is not found, login.html is used
   - in case of fatal errors, error.html is showed
4. **request for "/status" page**
   - if user is logged in, status.html is displayed
   - if user is not logged in, fstatus.html is displayed; if fstatus.html is not found, redirect.html is used to redirect to the login page

5. **request for '/logout' page**
   - if user is logged in, logout.html is displayed
   - if user is not logged in, flogout.html is displayed; if flogout.html is not found, redirect.html is used to redirect to the login page

Note: If it is not possible to meet a request using the pages stored on the router's FTP server, Error 404 is displayed

There are many possibilities to customize what the HotSpot authentication pages look like:

- The pages are easily modifiable. They are stored on the router's FTP server in the directory you choose for the respective HotSpot server profile.
- By changing the variables, which client sends to the HotSpot servlet, it is possible to reduce keyword count to one (username or password; for example, the client's MAC address may be used as the other value) or even to zero (License Agreement; some predefined values general for all users or client's MAC address may be used as username and password)
- Registration may occur on a different server (for example, on a server that is able to charge Credit Cards). Client's MAC address may be passed to it, so that this information need not be written in manually. After the registration, the server should change RADIUS database enabling client to log in for some amount of time.

To insert variable in some place in HTML file, the $(var_name) syntax is used, where the "var_name" is the name of the variable (without quotes). This construction may be used in any HotSpot HTML file accessed as '/', '/login', '/status' or '/logout', as well as any text or HTML (.txt, .htm or .html) file stored on the HotSpot server (with the exception of traffic counters, which are available in status page only, and **error**, **error-orig**, **chap-id**, **chap-challenge** and **popup** variables, which are available in login page only). For example, to show a link to the login page, following construction can be used:

```
<a href="$(link-login)">login</a>
```

Variables

All of the Servlet HTML pages use variables to show user specific values. Variable names appear only in the HTML source of the servlet pages - they are automatically replaced with the respective values by the HotSpot Servlet. For most variables there is an example of their possible value included in brackets. All the described variables are valid in all servlet pages, but some of them just might be empty at the time they are accesses (for example, there is no uptime before a user has logged in).

List of available variables

Common server variables:

- **`hostname`** - DNS name or IP address (if DNS name is not given) of the HotSpot Servlet ("hotspot.example.net")
- **`identity`** - RouterOS identity name ("MikroTik")
- **`login-by`** - authentication method used by user
- **`plain-passwd`** - a "yes/no" representation of whether HTTP-PAP login method is allowed ("no")
- **`server-address`** - HotSpot server address ("10.5.50.1:80")
- **`ssl-login`** - a "yes/no" representation of whether HTTPS method was used to access that servlet page ("no")
- **`server-name`** - HotSpot server name (set in the /ip hotspot menu, as the name property)

Links:

- **`link-login`** - link to login page including original URL requested ("http://10.5.50.1/login?dst=http://www.example.com/")
- **`link-login-only`** - link to login page, not including original URL requested ("http://10.5.50.1/login")
- **`link-logout`** - link to logout page ("http://10.5.50.1/logout")

- **link-status** - link to status page ("http://10.5.50.1/status")
- **link-orig** - original URL requested ("http://www.example.com/")

General client information:

- **domain** - domain name of the user ("example.com")
- **interface-name** - physical HotSpot interface name (in case of bridged interfaces, this will return the actual bridge port name)
- **ip** - IP address of the client ("10.5.50.2")
- **logged-in** - "yes" if the user is logged in, otherwise - "no" ("yes")
- **mac** - MAC address of the user ("01:23:45:67:89:AB")
- **trial** - a "yes/no" representation of whether the user has access to trial time. If users trial time has expired, the value is "no"
- **username** - the name of the user ("John")
- **host-ip** - client IP address from /ip hotspot host table

User status information:

- **idle-timeout** - idle timeout ("20m" or "" if none)
- **idle-timeout-secs** - idle timeout in seconds ("88" or "0" if there is such timeout)
- **limit-bytes-in** - byte limit for send ("1000000" or "---" if there is no limit)
- **limit-bytes-out** - byte limit for receive ("1000000" or "---" if there is no limit)
- **refresh-timeout** - status page refresh timeout ("1m30s" or "" if none)
- **refresh-timeout-secs** - status page refresh timeout in seconds ("90s" or "0" if none)
- **session-timeout** - session time left for the user ("5h" or "" if none)
- **session-timeout-secs** - session time left for the user, in seconds ("3475" or "0" if there is such timeout)
- **session-time-left** - session time left for the user ("5h" or "" if none)
- **session-time-left-secs** - session time left for the user, in seconds ("3475" or "0" if there is such timeout)
- **uptime** - current session uptime ("10h2m33s")
- **uptime-secs** - current session uptime in seconds ("125")

Traffic counters, which are available only in the status page:

- **bytes-in** - number of bytes received from the user ("15423")
- **bytes-in-nice** - user-friendly form of number of bytes received from the user ("15423")
- **bytes-out** - number of bytes sent to the user ("11352")
- **bytes-out-nice** - user-friendly form of number of bytes sent to the user ("11352")
- **packets-in** - number of packets received from the user ("251")
- **packets-out** - number of packets sent to the user ("211")
- **remain-bytes-in** - remaining bytes until limit-bytes-in will be reached ("337465" or "---" if there is no limit)
- **remain-bytes-out** - remaining bytes until limit-bytes-out will be reached ("124455" or "---" if there is no limit)

Miscellaneous variables:

- **session-id** - value of 'session-id' parameter in the last request
- **var** - value of 'var' parameter in the last request
- **error** - error message, if something failed ("invalid username or password")
- **error-orig** - original error message (without translations retrieved from errors.txt), if something failed ("invalid username or password")
- **chap-id** - value of chap ID ("\371")
- **chap-challenge** - value of chap challenge ("\357\015\330\013\021\234\145\245\303\253\142\246\133\175\375\316")

- **popup** - whether to pop-up checkbox ("true" or "false")
- **advert-pending** - whether an advertisement is pending to be displayed ("yes" or "no")
- **http-status** - allows to set http status code and message
- **http-header** - allows to add http header

RADIUS-related variables:

- **radius<id>** - show the attribute identified with <id> in text string form (in case RADIUS authentication was used; "" otherwise)
- **radius<id>u** - show the attribute identified with <id> in unsigned integer form (in case RADIUS authentication was used; "0" otherwise)
- **radius<id>-<vnd-id>** - show the attribute identified with <id> and vendor ID <vnd-id> in text string form (in case RADIUS authentication was used; "" otherwise)
- **radius<id>-<vnd-id>u** - show the attribute identified with <id> and vendor ID <vnd-id> in unsigned integer form (in case RADIUS authentication was used; "0" otherwise)

Working with variables

$(if <var_name>) statements can be used in theses pages. Following content will be included, if value of <var_name> will not be an empty string. It is an equivalent to $(if <var_name> != "") It is possible to compare on equivalence as well: $(if <var_name> == <value>) These statements have effect until $(elif <var_name>), $(else) or $(endif). In general case it looks like this:

```
some content, which will always be displayed
$(if username == john)
Hey, your username is john
$(elif username == dizzy)
Hello, Dizzy! How are you? Your administrator.
$(elif ip == 10.1.2.3)
You are sitting at that crappy computer, which is damn slow...
$(elif mac == 00:01:02:03:04:05)
This is an ethernet card, which was stolen few months ago...
$(else)
I don't know who you are, so lets live in peace.
$(endif)
other content, which will always be displayed
```

Only one of those expressions will be shown. Which one - depends on values of those variables for each client.

Redirects and custom Headers

Starting from RouterOS 5.12 there are 2 new hotspot html page variables:

- **http-status** - allows to set http status code and message
- **http-header** - allows to add http header

Example:

```
      $(if http-status == 302)Hotspot login required$(endif)
      $(if http-header == "Location")$(link-redirect)$(endif)
```

In case if $(link-redirect) will evaluate to "http://192.168.88.1/login", then HTTP response will look like:

```
HTTP/1.0 302 Hotspot login required
<regular HTTP headers>
```

```
Location: http://192.168.88.1/login
```

http-status syntax:

```
      $(if http-status == XYZ)HTTP_STATUS_MESSAGE$(endif)
```

- *XYZ* - status code, should be 3 decimal digits, first one must not be 0
- *HTTP_STATUS_MESSAGE* - any text, will follow status code in HTTP reply

In HTTP response it will be on first line and will look like:

```
HTTP/1.0 XYZ HTTP_STATUS_MESSAGE
```

http-header syntax:

```
      $(if http-header == HTTP_HEADER_NAME)HTTP_HEADER_VALUE$(endif)
```

- *HTTP_HEADER_NAME* - name of the HTTP header to add
- *HTTP_HEADER_VALUE* - value of HTTP header with name HTTP_HEADER_NAME

In HTTP response it will look like:

```
HTTP_HEADER_NAME: HTTP_HEADER_VALUE
```

All variables and conditional expressions within HTTP_HEADER_VALUE and HTTP_STATUS_MESSAGE are processed as usual.

In case multiple headers with the same name are added, then only the last one will be used (previous ones will be discarded). It allows to override regular HTTP headers (for example, Content-Type and Cache-Control).

Customizing Error Messages

All error messages are stored in the errors.txt file within the respective HotSpot servlet directory. You can change and translate all these messages to your native language. To do so, edit the errors.txt file. You can also use variables in the messages. All instructions are given in that file.

Multiple Versions of HotSpot Pages

Multiple HotSpot page sets for the same HotSpot server are supported. They can be chosen by user (to select language) or automatically by JavaScript (to select PDA/regular version of HTML pages).

To utilize this feature, create subdirectories in HotSpot HTML directory, and place those HTML files, which are different, in that subdirectory. For example, to translate everything in Latvian, subdirectory "lv" can be created with login.html, logout.html, status.html, alogin.html, radvert.html and errors.txt files, which are translated into Latvian. If the requested HTML page can not be found in the requested subdirectory, the corresponding HTML file from the main directory will be used. Then main login.html file would contain link to "/lv/login?dst=$(link-orig-esc)", which then displays Latvian version of login page: Latviski . And Latvian version would contain link to English version: English

Another way of referencing directories is to specify 'target' variable:

```
        <a href="$(link-login-only)?dst=$(link-orig-esc)&target=lv">Latviski</a>
        <a href="$(link-login-only)?dst=$(link-orig-esc)&target=%2F">English</a>
```

After preferred directory has been selected (for example, "lv"), all links to local HotSpot pages will contain that path (for example, $(link-status) = "http://hotspot.mt.lv/lv/status"). So, if all HotSpot pages reference links using "$(link-xxx)" variables, then no more changes are to be made - each client will stay within the selected directory all the time.

Misc

If you want to use HTTP-CHAP authentication method it is supposed that you include the **doLogin()** function (which references to the **md5.js** which must be already loaded) before the **Submit** action of the login form. Otherwise, CHAP login will fail.

The resulting password to be sent to the HotSpot gateway in case of HTTP-CHAP method, is formed MD5-hashing the concatenation of the following: chap-id, the password of the user and chap-challenge (in the given order)

In case variables are to be used in link directly, then they must be escaped accordingly. For example, in login page, **link** will not work as intended, if username will be "123&456=1 2". In this case instead of $(user), its escaped version must be used: $(user-esc): **link**. Now the same username will be converted to "123%26456%3D1+2", which is the valid representation of "123&456=1 2" in URL. This trick may be used with any variables, not only with $(username).

There is a boolean parameter "erase-cookie" to the logout page, which may be either "on" or "true" to delete user cookie on logout (so that the user would not be automatically logged on when he/she opens a browser next time.

Examples

With basic HTML language knowledge and the examples below it should be easy to implement the ideas described above.

- To provide predefined value as username, in login.html change:

```
<type="text" value="$(username)>
```

to this line:

```
<input type="hidden" name="username" value="hsuser">
```

(where hsuser is the username you are providing)

- To provide predefined value as password, in login.html change:

```
<input type="password">
```

to this line:

```
<input type="hidden" name="password" value="hspass">
```

(where hspass is the password you are providing)

- To send client's MAC address to a registration server in form of:

https://www.example.com/register.html?mac=XX:XX:XX:XX:XX:XX

change the Login button link in login.html to:

```
https://www.example.com/register.html?mac=$(mac)
```

(you should correct the link to point to your server)

- To show a banner after user login, in alogin.html after

$(if popup == 'true') add the following line:

```
open('http://www.example.com/your-banner-page.html', 'my-banner-name','');
```

(you should correct the link to point to the page you want to show)

- To choose different page shown after login, in login.html change:

```
<input type="hidden" name="dst" value="$(link-orig)">
```

to this line:

```
<input type="hidden" name="dst" value="http://www.example.com">
```

(you should correct the link to point to your server)

- To erase the cookie on logoff, in the page containing link to the logout (for example, in status.html) change:

```
open('$(link-logout)', 'hotspot_logout', ...
```

to this:

```
open('$(link-logout)?erase-cookie=on', 'hotspot_logout', ...
```

or alternatively add this line:

```
<input type="hidden" name="erase-cookie" value="on">
```

before this one:

```
<input type="submit" value="log off">
```

An another example is making HotSpot to authenticate on a remote server (which may, for example, perform creditcard charging):

- Allow direct access to the external server in walled-garden (either HTTP-based, or IP-based)
- Modify login page of the HotSpot servlet to redirect to the external authentication server. The external server should modify RADIUS database as needed

  Here is an example of such a login page to put on the HotSpot router (it is redirecting to https://auth.example.com/login.php, replace with the actual address of an external authentication server):

```
<html>
<title>...</title>
<body>
<form name="redirect" action="https://auth.example.com/login.php" method="post">
<input type="hidden" name="mac" value="$(mac)">
<input type="hidden" name="ip" value="$(ip)">
<input type="hidden" name="username" value="$(username)">
<input type="hidden" name="link-login" value="$(link-login)">
<input type="hidden" name="link-orig" value="$(link-orig)">
<input type="hidden" name="error" value="$(error)">
</form>
<script language="JavaScript">
<!--
      document.redirect.submit();
//-->
</script>
</body>
</html>
```

- The external server can log in a HotSpot client by redirecting it back to the original HotSpot servlet login page, specifying the correct username and password

Here is an example of such a page (it is redirecting to https://hotspot.example.com/login, replace with the actual address of a HotSpot router; also, it is displaying www.mikrotik.com after successful login, replace with what needed):

```
<html>
<title>Hotspot login page</title>
<body>
<form name="login" action="https://hotspot.example.com/login" method="post">
<input type="text" name="username" value="demo">
<input type="password" name="password" value="none">
<input type="hidden" name="domain" value="">
<input type="hidden" name="dst" value="http://www.mikrotik.com/">
<input type="submit" name="login" value="log in">
</form>
</body>
</html>
```

- Hotspot will ask RADIUS server whether to allow the login or not. If not allowed, alogin.html page will be displayed (it can be modified to do anything). If not allowed, flogin.html (or login.html) page will be displayed, which will redirect client back to the external authentication server.

Note: as shown in these examples, HTTPS protocol and POST method can be used to secure communications.

Firewall customizations

Summary

Apart from the obvious dynamic entries in the /ip hotspot submenu itself (like hosts and active users), some additional rules are added in the firewall tables when activating a HotSpot service. Unlike RouterOS version 2.8, there are relatively few firewall rules added in the firewall as the main job is made by the one-to-one NAT algorithm.

NAT

From **/ip firewall nat print dynamic** command, you can get something like this (comments follow after each of the rules):

```
 0 D chain=dstnat action=jump jump-target=hotspot hotspot=from-client
```

Putting all HotSpot-related tasks for packets from all HotSpot clients into a separate chain.

```
 1 I chain=hotspot action=jump jump-target=pre-hotspot
```

Any actions that should be done before HotSpot rules apply, should be put in the pre-hotspot chain. This chain is under full administrator control and does not contain any rules set by the system, hence the invalid jump rule (as the chain does not have any rules by default).

```
 2 D chain=hotspot action=redirect to-ports=64872 dst-port=53 protocol=udp
 3 D chain=hotspot action=redirect to-ports=64872 dst-port=53 protocol=tcp
```

Redirect all DNS requests to the HotSpot service. The 64872 port provides DNS service for all HotSpot users. If you want HotSpot server to listen also to another port, add rules here the same way, changing dst-port property.

```
 4 D chain=hotspot action=redirect to-ports=64873 hotspot=local-dst dst-port=80
     protocol=tcp
```

Redirect all HTTP login requests to the HTTP login servlet. The 64873 is HotSpot HTTP servlet port.

```
 5 D chain=hotspot action=redirect to-ports=64875 hotspot=local-dst dst-port=443
     protocol=tcp
```

Redirect all HTTPS login requests to the HTTPS login servlet. The 64875 is HotSpot HTTPS servlet port.

```
 6 D chain=hotspot action=jump jump-target=hs-unauth hotspot=!auth protocol=tcp
```

All other packets except DNS and login requests from unauthorized clients should pass through the hs-unauth chain.

```
 7 D chain=hotspot action=jump jump-target=hs-auth hotspot=auth protocol=tcp
```

And packets from the authorized clients - through the hs-auth chain.

```
 8 D ;;; www.mikrotik.com
     chain=hs-unauth action=return dst-address=66.228.113.26 dst-port=80 protocol=tcp
```

First in the **hs-unauth** chain is put everything that affects TCP protocol in the `/ip hotspot walled-garden ip` submenu (i.e., everything where either protocol is not set, or set to TCP). Here we are excluding www.mikrotik.com from being redirected to the login page.

```
 9 D chain=hs-unauth action=redirect to-ports=64874 dst-port=80 protocol=tcp
```

All other HTTP requests are redirected to the Walled Garden proxy server which listens the 64874 port. If there is an allow entry in the `/ip hotspot walled-garden` menu for an HTTP request, it is being forwarded to the destination. Otherwise, the request will be automatically redirected to the HotSpot login servlet (port 64873).

```
10 D chain=hs-unauth action=redirect to-ports=64874 dst-port=3128 protocol=tcp
11 D chain=hs-unauth action=redirect to-ports=64874 dst-port=8080 protocol=tcp
```

HotSpot by default assumes that only these ports may be used for HTTP proxy requests. These two entries are used to "catch" client requests to unknown proxies (you can add more rules here for other ports). I.e., to make it possible for the clients with unknown proxy settings to work with the HotSpot system. This feature is called "Universal Proxy". If it is detected that a client is using some proxy server, the system will automatically mark that packets with the http hotspot mark to work around the unknown proxy problem, as we will see later on. Note that the port used (64874) is the same as for HTTP requests in the rule #9 (so both HTTP and HTTP proxy requests are processed by the same code).

```
12 D chain=hs-unauth action=redirect to-ports=64875 dst-port=443 protocol=tcp
```

HTTPS proxy is listening on the 64875 port.

```
13 I chain=hs-unauth action=jump jump-target=hs-smtp dst-port=25 protocol=tcp
```

Redirect for SMTP protocol may also be defined in the HotSpot configuration. In case it is, a redirect rule will be put in the hs-smtp chain. This is done so that users with unknown SMTP configuration would be able to send their mail through the service provider's (your) SMTP server instead of going to the [possibly unavailable outside their network of origin] SMTP server users have configured on their computers. The chain is empty by default, hence the invalid jump rule.

```
14 D chain=hs-auth action=redirect to-ports=64874 hotspot=http protocol=tcp
```

Providing HTTP proxy service for authorized users. Authenticated user requests may need to be subject to transparent proxying (the "Universal Proxy" technique and advertisement feature). This http mark is put automatically on the HTTP proxy requests to the servers detected by the HotSpot HTTP proxy (the one that is listening on the 64874 port) as HTTP proxy requests for unknown proxy servers. This is done so that users that have some proxy settings would use the HotSpot gateway instead of the [possibly unavailable outside their network of origin] proxy server users have configured in their computers. This mark is also applied when advertisement is due to be shown to the user, as well as on any HTTP requests done form the users whose profile is configured to transparently proxy their requests.

```
15 I chain=hs-auth action=jump jump-target=hs-smtp dst-port=25 protocol=tcp
```

Providing SMTP proxy for authorized users (the same as in rule #13).

Packet Filtering

From **/ip firewall filter print dynamic** command, you can get something like this (comments follow after each of the rules):

```
 0 D chain=forward action=jump jump-target=hs-unauth hotspot=from-client,!auth
```

Any packet that traverse the router from an unauthorized client will be sent to the **hs-unauth** chain. The hs-unauth implements the IP-based Walled Garden filter.

```
 1 D chain=forward action=jump jump-target=hs-unauth-to hotspot=to-client,!auth
```

Everything that comes to clients through the router, gets redirected to another chain, called **hs-unauth-to**. This chain should reject unauthorized requests to the clients.

```
 2 D chain=input action=jump jump-target=hs-input hotspot=from-client
```

Everything that comes from clients to the router itself, gets to yet another chain, called **hs-input**.

```
 3 I chain=hs-input action=jump jump-target=pre-hs-input
```

Before proceeding with [predefined] dynamic rules, the packet gets to the administratively controlled **pre-hs-input** chain, which is empty by default, hence the invalid state of the jump rule.

```
 4 D chain=hs-input action=accept dst-port=64872 protocol=udp
 5 D chain=hs-input action=accept dst-port=64872-64875 protocol=tcp
```

Allow client access to the local authentication and proxy services (as described earlier).

```
 6 D chain=hs-input action=jump jump-target=hs-unauth hotspot=!auth
```

All other traffic from unauthorized clients to the router itself will be treated the same way as the traffic traversing the routers.

```
 7 D chain=hs-unauth action=return protocol=icmp
 8 D ;;; www.mikrotik.com
     chain=hs-unauth action=return dst-address=66.228.113.26 dst-port=80 protocol=tcp
```

Unlike NAT table where only TCP-protocol related Walled Garden entries were added, in the packet filter **hs-unauth** chain is added everything you have set in the **/ip hotspot walled-garden ip** menu. That is why although you have seen only one entry in the NAT table, there are two rules here.

```
 9 D chain=hs-unauth action=reject reject-with=tcp-reset protocol=tcp
10 D chain=hs-unauth action=reject reject-with=icmp-net-prohibited
```

Everything else that has not been while-listed by the Walled Garden will be rejected. Note usage of TCP Reset for rejecting TCP connections.

```
11 D chain=hs-unauth-to action=return protocol=icmp
12 D ;;; www.mikrotik.com
     chain=hs-unauth-to action=return src-address=66.228.113.26 src-port=80 protocol=tcp
```

Same action as in rules #7 and #8 is performed for the packets destined to the clients (chain **hs-unauth-to**) as well.

```
13 D chain=hs-unauth-to action=reject reject-with=icmp-host-prohibited
```

Reject all packets to the clients with ICMP reject message.

[Top | Back to Content **]**

Manual:IP/Hotspot

HotSpot

The MikroTik HotSpot Gateway provides authentication for clients before access to public networks .

HotSpot Gateway features:

- different authentication methods of clients using local client database on the router, or remote RADIUS server;
- users accounting in local database on the router, or on remote RADIUS server;
- walled-garden system, access to some web pages without authorization;
- login page modification, where you can put information about the company;
- automatic and transparent change any IP address of a client to a valid address;

Sub Categories

List of reference sub-pages **Case studies** **List of examples**

<splist showparent=yes />

HotSpot Setup

The simplest way to setup HotSpot server on a router is by **`/ip hotspot setup`** command. Router will ask to enter parameters required to successfully set up HotSpot. When finished, default configuration will be added for HotSpot server.

```
[admin@MikroTik] /ip hotspot> setup
Select interface to run HotSpot on

hotspot interface: ether3
Set HotSpot address for interface

local address of network: 10.5.50.1/24
masquerade network: yes
Set pool for HotSpot addresses

address pool of network: 10.5.50.2-10.5.50.254
```

```
Select hotspot SSL certificate

select certificate: none
Select SMTP server

ip address of smtp server: 0.0.0.0
Setup DNS configuration

dns servers: 10.1.101.1
DNS name of local hotspot server

dns name: myhotspot
Create local hotspot user

name of local hotspot user: admin
password for the user:
[admin@MikroTik] /ip hotspot>
```

What was created:

```
[admin@MikroTik] /ip hotspot> print
Flags: X - disabled, I - invalid, S - HTTPS
 #   NAME        INTERFACE       ADDRESS-POOL       PROFILE       IDLE-TIMEOUT
 0   hotspot1    ether3          hs-pool-3          hsprof1       5m
[admin@MikroTik] /ip hotspot>
[admin@MikroTik] /ip pool> print
 # NAME                                      RANGES
 0 hs-pool-3                                 10.5.50.2-10.5.50.254
[admin@MikroTik] /ip pool> /ip dhcp-server
[admin@MikroTik] /ip dhcp-server> print
Flags: X - disabled, I - invalid
 #   NAME      INTERFACE    RELAY           ADDRESS-POOL    LEASE-TIME ADD-ARP
 0   dhcp1     ether3                       hs-pool-3       1h
[admin@MikroTik] /ip dhcp-server> /ip firewall nat
[admin@MikroTik] /ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0 X ;;; place hotspot rules here
     chain=unused-hs-chain action=passthrough

 1   ;;; masquerade hotspot network
     chain=srcnat action=masquerade src-address=10.5.50.0/24
[admin@MikroTik] /ip firewall nat>
```

Parameters asked during setup process

Parameter	Description
hotspot interface (*string*; Default: **allow**)	Interface name on which to run HotSpot. To run HotSpot on a bridge interface, make sure public interfaces are not included to the bridge ports.
local address of network (*IP*; Default: **10.5.50.1/24**)	HotSpot gateway address
masquerade network (*yes* | *no*; Default: **yes**)	Whether to masquerade HotSpot network, when **yes** rule is added to */ip firewall nat* with *action=masquerade*
address pool of network (*string*; Default: **yes**)	Address pool for HotSpot network, which is used to change user IP address to a valid address. Useful if providing network access to mobile clients that are not willing to change their networking settings.
select certificate (*none* | *import-other-certificate*; Default:)	Choose SSL certificate, when HTTPS authorization method is required.
ip address of smtp server (*IP*; Default: **0.0.0.0**)	IP address of the SMTP server, where to redirect HotSpot's network SMTP requests (25 TCP port)
dns servers (*IP*; Default: **0.0.0.0**)	DNS server addresses used for HotSpot clients, configuration taken from */ip dns* menu of the HotSpot gateway
dns name (*string*; Default: **""**)	domain name of the HotSpot server, full quality domain name is required, for example www.example.com
name of local hotspot user (*string*; Default: **"admin"**)	username of one automatically created HotSpot user, added to */ip hotspot user*
password for the user' (*string*; Default:)	Password for automatically created HotSpot user

ip hotspot

Menu is designed to manage HotSpot servers of the router. It is possible to run HotSpot on Ethernet, wireless, VLAN and bridge interfaces. One HotSpot server is allowed per interface. When HotSpot is configured on bridge interface, set HotSpot interface as *bridge* interface not as *bridge port*, do not add public interfaces to bridge ports. You can add HotSpot servers manually to */ip hotspot* menu, but it is advised to run */ip hotspot setup*, that adds all necessary settings.

- **name** (text) : HotSpot server's name or identifier
- **address-pool** (name **/** none; default: *none*) : address space used to change HotSpot client *any* IP address to a valid address. Useful for providing public network access to mobile clients that are not willing to change their networking settings
- **idle-timeout** (time **/** none; default: *5m*) : period of inactivity for unauthorized clients. When there is no traffic from this client (literally client computer should be switched off), once the timeout is reached, user is dropped from the HotSpot host list, its used address becomes available
- **interface** (name of interface) : interface to run HotSpot on
- **addresses-per-mac** (integer **/** unlimited; default: 2) : number of IP addresses allowed to be bind with the MAC address, when multiple HotSpot clients connected with one MAC-address
- **profile** (name; default: *default*) - HotSpot server default HotSpot profile, which is located in */ip hotspot profile*

ip hotspot active

HotSpot active menu shows all clients authenticated in HotSpot, menu is informational it is not possible to change anything here.

- **server** (read-only; name) : HotSpot server name client is logged in
- **user** (read-only; name) : name of the HotSpot user
- **domain** (read-only; text) : domain of the user (if split from username), parameter is used only with RADIUS authentication
- **address** (read-only; IP address) : IP address of the HotSpot user
- **mac-address** (read-only; MAC-address) : MAC-address of the HotSpot user
- **login-by** (read-only; multiple choice: cookie **/** http-chap **/** http-pap **/** https **/** mac **/** mac **/** trial) : authentication method used by HotSpot client
- **uptime** (read-only; time) : current session time of the user, it is showing how long user has been logged in
- **idle-time** (read-only; time) : the amount of time user has been idle
- **session-time-left** (read-only; time) : the exact value of session-time, that is applied for user. Value shows how long user is allowed to be online to be logged of automatically by **uptime** reached
- **idle-timeout** (read-only; time) : the exact value of the user's idle-timeout
- **keepalive-timeout** (read-only; time) : the exact value of the keepalive-timeout, that is applied for user. Value shows how long host can stay out of reach to be removed from the HotSpot
- **limit-bytes-in** (read-only; integer) : value shows how many bytes received from the client, option is active when the appropriate parameter is configured for HotSpot user
- **limit-bytes-out** (read-only; integer) : value shows how many bytes send to the client, option is active when the appropriate parameter is configured for HotSpot user
- **limit-bytes-total** (read-only; integer) : value shows how many bytes total were send/received from client, option is active when the appropriate parameter is configured for HotSpot user

ip hotspot host

Host table lists all computers connected to the HotSpot server. Host table is informational and it is not possible to change any value there

- **mac-address** (read-only; MAC-address) : HotSpot user MAC-address
- **address** (read-only; IP address) : HotSpot client original IP address
- **to-address** (read-only; IP address) : New client address assigned by HotSpot, it might be the same as original **address**
- **server** (read-only; name) : HotSpot server name client is connected to
- **bridge-port** (read-only; name) : /interface bridge port client connected to, value is unknown when HotSpot is not configured on the bridge
- **uptime** (read-only; time) : value shows how long user is online (connected to the HotSpot)
- **idle-time** (read-only; time) : time user has been idle
- **idle-timeout** (read-only; time) : value of the client idle-timeout (unauthorized client)
- **keeaplive-timeout** (read-only; time) : keepalive-timeout value of the unauthorized client
- **bytes-in** (read-only; integer) : amount of bytes received from unauthorized client
- **packet-in** (read-only; integer) : amount of packets received from unauthorized client
- **bytes-out** (read-only; integer) : amount of bytes send to unauthorized client
- **packet-out** (read-only; integer) : amount of packets send to unauthorized client

IP Bindings

Sub-menu: `/ip hotspot ip-binding`

IP-Binding HotSpot menu allows to setup static One-to-One NAT translations, allows to bypass specific HotSpot clients without any authentication, and also allows to block specific hosts and subnets from HotSpot network

Property	Description
address (*IP Range*; Default: **""**)	The original IP address of the client
mac-address (*MAC*; Default: **""**)	MAC address of the client
server (*string* | *all*; Default: **"all"**)	Name of the HotSpot server. • **all** - will be applied to all hotspot servers
to-address (*IP*; Default: **""**)	New IP address of the client, translation occurs on the router (client does not know anything about the translation)
type (*blocked* | *bypassed* | *regular*; Default: **""**)	Type of the IP-binding action • **regular** - performs One-to-One NAT according to the rule, translates **address** to **to-address** • **bypassed** - performs the translation, but excludes client from login to the HotSpot • **blocked** - translation is not performed and packets from host are dropped

Cookies

Sub-menu: `/ip hotspot cookie`

Menu contains all cookies sent to the HotSpot clients, which are authorized by cookie method, all the entries are read-only.

Property	Description
domain (*string*)	Domain name (if split from username)
expires-in (*time*)	How long the cookie is valid
mac-address (*MAC*)	Client's MAC-address
user (*string*)	HotSpot username

[Top | Back to Content]

Manual:Virtualization

Applies to RouterOS: 3, v4

RouterOS has three different Virtualization implementations. Choose your topic:

- Metarouter
- Xen
- Kvm

Metarouter

Metarouter is created by MikroTik and currently is supported only on RouterBOARD 4xx series (mips-be) and RB1000 series (powerpc). Currently Metarouter can only create RouterOS virtual machines.

We are planning to add more features to Metarouter, so that it will even exceed Xen in functionality. New hardware support will also be added to Metarouter

Xen

Xen is based on the Linux Xen Virtual machine project, and current RouterOS implementation is supported only on RouterOS X86 systems (PCs). Xen can create Virtual machines of different Operating Systems that supports Xen.

Kvm

Kvm is based on Linux Kvm virtualization software and requires your CPU to support virtualization. Kvm is available only on x86 systems.

Usage Examples

The following are just a few of possible scenarios where virtual machines could be used (some of these currently are possible only in Xen, but Metarouter features will be expanded to allow even more functionality):

In the datacenter

- consolidate a number of routers on one hardware platform
- consolidate routing services and higher levels services such a VOIP switches in the same box
- use a guest machine on top of a router for custom features such as accounting, LDAP or legacy networking
- redundant routers much easier and cheaper to have available in case of crashed systems

In the hosting center

- use RouterOS and extensive networking features as a host with a server (mail, http, ftp...) running as guest or multiple guest virtual machines
- offer virtual routers with VPN solutions that give a network administrator customer his own router on a highspeed backbone to make any kind of tunneled intranet or simply VPN access system

At the wireless ISP client site

- set up two isolated routers and set the wireless control only for the router controlled by the WISP while the Ethernet side router is fully under the clients control

At multiclient sites (such as office buildings)

- in locations serving multiple clients by Ethernet from one backbone connection (wired or wireless), give each customer control over his own isolated virtual router

For network planning and testing

- build a virtual network on one box with the same topography as a planned network and test the configurations so that the fine tuning of the configurations can be done in the lab and not in the field, simulate and monitor the network with advanced scripting and The Dude network monitor utility

In custom applications

- develop your own programs (and even Linux distributions) that can be installed on MikroTik supported platforms with minimum difficulty as software patches and virtual drivers are provided for guest systems
- use low cost RouterBOARD embedded systems easily with your own Linux and the advantage that it will work across all RouterBOARDS with the same CPU

Manual:KVM

Applies to RouterOS: v4.3+ on x86

Overview

Kernel-based Virtual Machine (KVM) is the method to run multiple guest operating systems on one RouterOS host. KVM can be used only on x86 machines that have CPU with virtualization support .

Requirements

KVM requires **Intel VT-x** or **AMD-V** CPU virtualization support. Here [1] you can find a list of supported CPUs, for more detailed information look on vendor's web site.

Each guest requires at least 16 MB of RAM and sufficient storage space on image file. Once image file have been created, its size cannot be increased.

KVM support in RouterOS is enabled if *kvm* package is installed.

Where it can be used?

Virtual Router is useful to allow clients or lower-privilege users access their own 'router' and adjust configure as they like without the need for a second hardware.

For example; a WISP can create a virtual router for the clients ethernet port allowing them to define their own firewall settings, while leaving the WISP's wireless settings untouched.

Another useful method is to run guest OS that supports functionality which is not available in RouterOS, for example, Intrusion detection (SNORT), Asterisk or Squid web proxy.

It can also be used as test environment. it is possible to create virtual network within one x86 machine very similar to real network and test how RouterOS behaves before implementing the setup in your production network.

Creating KVM Guest

Before creating KVM guest we need image file. RouterOS has built in commands to make and modify RouterOS image easily without external tools.

```
/kvm make-routeros-image file-name=ros1.img file-size=128
```

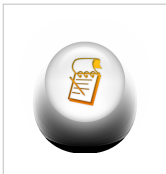
We can proceed with Guest configuration when disk image is created.

```
/kvm add name=ROS memory=128MiB cpu-count=2 disabled=no disk-images=hda:ros1.img \
    initrd="" kernel="" kernel-cmdline="console=ttyS0"
```

As you noticed **initrd** and **kernel** properties are empty, which means that hosts kernel and initrd is used.

For example, to add guest without SMP support we can explicitly set initrd and kernel:

```
/kvm add name=ROS memory=128MiB cpu-count=2 disabled=no disk-images=hda:ros1.img \
    initrd=/boot/initrd.rgz kernel=/boot/vmlinuz kernel-cmdline="console=ttyS0"
```



Note: Leaving **initrd** and **kernel** properties empty is dangerous if Host and Guest will be running different RouterOS versions . Guests other than RouterOS also can break if you leave these values empty.

KVM Guest when created is not automatically started. We must start it manually

```
[admin@proxy] /kvm> start ROS;
```

```
[admin@proxy] /kvm> print
Flags: X - disabled
 0   name="ROS" cpu-count=2 memory=128MiB disk-images=hda:ros1.img kernel="/boot/vmlinuz"
      kernel-cmdline="" initrd="/boot/initrd.rgz" vnc-server=0.0.0.0:0 snapshot=no  state=running
[admin@proxy] /kvm>
```

Adding Interfaces

Lets add to our previously created Virtual Router one interface.

```
[admin@proxy] /kvm interface> add virtual-machine=ROS type=dynamic
[admin@proxy] /kvm interface> print
Flags: X - disabled, A - active
 #   VIRTUAL-MACHINE            INTERFACE                  TYPE         VM-MAC-ADDRESS
 0   ROS                                                   dynamic      02:D9:52:31:11:CC
[admin@proxy] /kvm interface>
```

In this case dynamic type is used which creates dynamic virtual interface on the host:

```
[admin@proxy] /interface virtual-ethernet> print
Flags: D - dynamic, X - disabled, R - running
 #      NAME                              MTU   ARP         MAC-ADDRESS
 0 D R tap1                               1500  enabled     02:3F:9F:AE:10:34
[admin@proxy] /interface virtual-ethernet>
```

Note: Add and remove interfaces only when KVM guest is shut-down, stopped or disabled. Making changes to running guest may lead to host system crash.

If mac addresses are not specified when creating virtual interfaces, addresses are generated automatically. Generate MAC addresses will be in form of 02:XX:XX:XX:XX:XX. For *static* interfaces this address will not change during use of guest, for *dynamic* interface will change every time dynamic interface is created.

More information about virtual interfaces are in virtual-ethernet manual

Connecting to the virtual machine

There are two ways how to connect to KVM Guest:

- virtual console;
- vnc.

Console

To connect using console:

```
[admin@proxy] /kvm> console ROS
```

You will see your newly added virtual interface here:

```
[admin@mr0] > interface print
 Flags: D - dynamic, X - disabled, R - running, S - slave
 #     NAME                                                TYPE               MTU
 0  R  ether1                                              ether              1500
```

To disconnect from the metarouter virtual machine console, hit **CTRL + A** and then **Q** to Quit back to your **Host** console (if you are using minicom, hit **CTRL + A** twice):

```
[admin@MikroTik] >
[Q - quit connection]       [B - send break]
[A - send Ctrl-A prefix]    [R - autoconfigure rate]

 Q

 Welcome back!
```

VNC

Before connecting with VNC client guest needs some configuration changes.

```
[admin@proxy] /kvm> print
Flags: X - disabled
 0   name="ROS" cpu-count=2 memory=128MiB disk-images=hda:ros1.img kernel="/boot/vmlinuz"
      kernel-cmdline="" initrd="/boot/initrd.rgz" vnc-server=0.0.0.0:0 snapshot=no state=running
[admin@proxy] /kvm> shut-down 0
[admin@proxy] /kvm> set 0 vnc-server=10.5.100.99:1
[admin@proxy] /kvm> start 0
```

```
[admin@proxy] /kvm> print
Flags: X - disabled
```

```
 0   name="ROS" cpu-count=2 memory=128MiB disk-images=hda:ros1.img kernel="/boot/vmlinuz"
      kernel-cmdline="" initrd="/boot/initrd.rgz" vnc-server=10.5.100.99:1 snapshot=no
      state=running
[admin@proxy] /kvm>
```

VNC servers address in this case is the address on the host reachable from remote locations. Address is followed by screen number.

Now we can try to connect from remote location:

```
mrz@bumba:/$ vncviewer 10.5.100.99:1
```

Configuring a virtual network

Right now you saw that the virtual interface is visible in the **Host** *Interfaces* menu as *tap1* and also in the **guest** *interfaces* menu as *ether1*. You can add an IP address on both interfaces, and set up networking. Creating a bridge between the virtual interface and a physical interface allows traffic to pass.

As an example lets make three virtual routers connected to each other on the same broadcast domain.

Create images and guests:

```
/kvm
make-routeros-image file-name=R1.img file-size=64
make-routeros-image file-name=R2.img file-size=64
make-routeros-image file-name=R3.img file-size=64

add name=R1 disk-image=hda:R1.img
add name=R2 disk-image=hda:R2.img
add name=R3 disk-image=hda:R3.img
```

Create a bridge interface which will simulate broadcast domain and add virtual interfaces:

```
/interface bridge
add name=kvm_bridge

/kvm interface
add virtual-machine=R1 type=dynamic dynamic-bridge=kvm_bridge
add virtual-machine=R2 type=dynamic dynamic-bridge=kvm_bridge
add virtual-machine=R3 type=dynamic dynamic-bridge=kvm_bridge
```

Now we can start virtual machines and verify if dynamic interfaces are created:

```
[admin@proxy] /kvm> start [find]
```

```
[admin@proxy] > /interface virtual-ethernet print
Flags: D - dynamic, X - disabled, R - running
 #     NAME                              MTU   ARP        MAC-ADDRESS
 0 D R tap2                              1500  enabled    02:20:94:67:D6:D5
 1 D R tap3                              1500  enabled    02:95:EE:EA:43:FF
 2 D R tap4                              1500  enabled    02:05:7E:4B:86:F9
[admin@proxy] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE               BRIDGE                    PRIORITY PATH-COST  HORIZON
```

```
 0  D tap2                    kvm_bridge              0x80     10         none
 1  D tap3                    kvm_bridge              0x80     10         none
 2  D tap4                    kvm_bridge              0x80     10         none
[admin@proxy] >
```

Now we can connect with console to each of guests and set up ip addresses from the same network and verify reachability. R1

```
[admin@proxy] > /kvm console R1

[Ctrl-A is the prefix key]

MikroTik 5.0rc8
MikroTik Login: admin
Password:

 [admin@MikroTik] > /ip address add address=192.168.1.1/24 interface=ether1
```

R2

```
<pre>
[admin@proxy] > /kvm console R2

[Ctrl-A is the prefix key]

MikroTik 5.0rc8
MikroTik Login: admin
Password:

 [admin@MikroTik] > /ip address add address=192.168.1.2/24 interface=ether1
```

R3

```
<pre>
[admin@proxy] > /kvm console R1

[Ctrl-A is the prefix key]

MikroTik 5.0rc8
MikroTik Login: admin
Password:

[admin@MikroTik] > /ip address add address=192.168.1.3/24 interface=ether1
[admin@MikroTik] > /ping 192.168.1.1
HOST                                     SIZE  TTL TIME  STATUS
192.168.1.1                              56    64  11ms
192.168.1.1                              56    64  2ms
    sent=2 received=2 packet-loss=0% min-rtt=2ms avg-rtt=6ms max-rtt=11ms

[admin@MikroTik] > /ping 192.168.1.2
```

```
HOST                                     SIZE  TTL TIME  STATUS
192.168.1.2                              56    64  12ms
    sent=1 received=1 packet-loss=0% min-rtt=12ms avg-rtt=12ms max-rtt=12ms
```

Removing KVM guest

KVM guest has two parts in RouterOS - configuration (kvm, virtual-ethernet, /kvm interface) and image file (/file). If image file is removed, but KVM guest is still running, then file will be removed from file menu but still exist until guest is shut-down or disabled, at that moment file will be removed and storage space returned to available storage on the router.

Additional information

Information useful for running KVM guests

Host shutdown

When host is shutting down each guest receives shut-down notification and are give 10 seconds to shut down. After time-out value is reached, guests are killed.

Host and guest update

When new version of RouterOS is updated to host system and you have RouterOS guest with *initrd* and *kernel* fields empty, it is good practice to update guest first (even it it does not boot up at current host versions. Then update host and see if guests are running. After guest update incompatibilities between host kernel and guest drivers might prevent guest from booting up properly.

Reference

General

Sub-menu: `/kvm`

KVM Guest Properties

To add new KVM guest you will have to issue command *add* under */kvm* menu with attributes as follows:

Property	Desciption
comment (*text, default: ''*)	to add simple *text* description of the KVM guest
cpu-count (*1 .. 32, default: 1*)	available count of processing cores for guest. Allowed values are [1..32]
disabled (*yes | no, default: no*)	to set guest state after creation, values: *yes* or *no*

disk-images (*list of images used in guest*)	list of image assignment to drives for guest OS. If type will be set to ***cdrom*** then guest will automatically boot from that, instead of any other drive configured in this field. It can be single drive specified
	`disk-images=hda:ros.img`
	or it can be comma seperated list:
	`disk-images=hda:system.img,hdb:swap.img`

initrd (*path*)	path to initrd file, can be left empty if running RouterOS as guest
kernel (*path*)	path to kernel image file, if using RouterOS image created on host this field can be left empty

kernel-cmdline (*text*)	parameters that are passed to kernel, it is space separated string.
memory (*integer default:32*)	to set up amount of memory that is available to KVM guest
name (*text*)	name of KVM guest that it will be accessible though the system
snapshot (*yes | no*)	will try to run virtual machine with image file in read-only mode.
vnc-server-address (*IP address*)	address to bind VNC server port that will connect to guest virtual screen. If left empty it will bind to all IP addresses. If address set is not ready at the moment when guest is started then system will automatically attempt to start guest for the next 20 seconds. If IP address to bind VNC does not become available in that time automatic start of guest will fail and guest will not be started. IP address is considered unavailable if either address or interface address is assigned to is invalid or does not exist.
vnc-server-display (*number (0..99) default:0*)	will try to run virtual machine with image file in read-only
copy-from (*number*)	use configuration from already existing KVM guest

Warning: vnc-server attribute has been changed since RouterOS 5.0. in older versions instead of vnc-server-address and vnc-server-display was used combine attribute named vnc-server <IP address>:<display number>

Note: If start of guest failed for the first time, then next 20 seconds KVM will attempt to start guest. After 20 seconds it will fail and guest will stay in stopped state.

States of KVM guest

This field is read-only and is set by RouterOS. These are possible values that can be set:

- *stopped* - KVM guest is not running, either successful shut-down or disabled.
- *stopping* - KVM guest is shutting down
- *starting* - KVM guest is starting
- *running* - KVM guest has started successfully and is executing guest operating system
- *restarting* - KVM guest is reloading its guest operating system
- *failed* - KVM guest has encountered an error and is not operational.
- *image-busy* - image file set in configuration is already in use by other KVM guest entry
- *no-kernel-or-initrd* - initrd or kernel was not found in files set in configuration, mentioned files could not be found or no values in those fields where set
- *no-disk-image* - either disk image was not found or disk image was not set in configuration.
- *kernel-extract-failed* - when in guest configuration field *kernel* is left empty and and KVM cannot extract kernel from image file supplied
- *vnc-cant-bind* - vnc server for guest cannot bind to setting specified in *vnc-server-address* and/or *vnc-server-display*

KVM commands

Sub-menu allows to manage KVM guests on RouterOS host.

Command	Desciption
add	Create new KVM guest entry
comment	Set comment for KVM guest entry
console	to connect to KVM guest console display
continue	resume KVM guest if it was paused
disable	change global state of KVM guest. If enabled KVM guest will be started when RouterOS boots. KVM guest cannot change
edit	edit selected value of KVM guest entry
enable	change KVM guest global state to enable operation of KVM guest. If guest where disabled before - KVM guest is automatically started.
export	Print or save an export script that can be used to restore configuration of current sub-menu, KVM guest configuration, image files will not be saved
find	Find items by value
get	Gets value of item's property
make-routeros-image	creates RouterOS image from current installation installed on the router with no configuration. It is advised to create Image file larger than minimal, so you are able to upload new package files and upgrade/update RouterOS installation. Also, all the additional files created in KVM guest will be stored in file image. This image file is not connected to host RouterOS and user is able to run different RouterOS versions on host and guest. This command will create RAW image file containing RouterOS installation. parameters: • `file-name` - name of ROS image file; • `file-size` - image size in Meba Bytes; • `configuration-script` - file name where configuration script is located;
pause	suspend operation of KVM guest
print	Print values of item properties
reboot	issue ACPI ***shut-down*** command to KVM guest, if guest does not support ACPI, command have no effect. After KVM guest is shut-downed it will be automatically started by host when shut down is complete.
reconfigure-routeros-image	sets up default configuration for RouterOS image. Parameters: • `file-name` - name of ROS image to be reconfigured; • `configuraton-script` - file name where configuration script is located; • `configuration-string` - string containing ROS commands to be configured on ROS image.
remove	Remove item
set	Change item properties
shut-down	issue ACPI shut-down command to KVM guest, if guest does not support ACPI, command have no effect.
start	to start KVM guest

Interface

Sub-menu: `/kvm interface`

Property	Desciption
comment (*text*)	description of interface
disabled (*yes|no, default: no*)	state of interface after creation
host-mac-address (*MAC Address, default: generated*)	MAC address of virtual interface that host will use
model (*virto | e1000 | pcnet, default: virtio*)	mode of virtual interface. Available options are: • *virtio - default* value. Fastest available option, should be chosen if no other problems are encountered • *e1000* - emulates card that uses e1000 driver. *This option where added for compatibility with some guest operating systems that where not able to communicate with host RouterOS if* virtio *interface model where used.* • *pcnet* - emulates card that uses pcnet driver. *This option where added for compatibility with some guest operating systems that where not able to communicate with host RouterOS if* virtio *interface model where used.*
vm-mac-address (*MAC Address, default: generated*)	MAC address of virtual interface that guest will use
copy-from (*number*)	use configuration from existing virtual interface
dynamic-bridge (*interface name, default: none*)	if set, ***dynamic*** interface will be automatically added as port to bridge interface
interface	is set for ***static*** interface, to assign it to already created virtual-ethernet interface
type (*dynamic | static, default: static*)	to set if interface is either *static* or *dynamic*. • *dynamic* interface will add virtual-ethernet automatically when virtual machine starts. • *static* interface have to have created virtual-ethernet interface at the time of creation of the entry.
virtual-machine (*KVM machine name, must be set)*	name of virtual machine this interface will be assigned to

References

[1] http://en.wikipedia.org/wiki/X86_virtualization

Manual:Xen

Xen Virtualization Overview

XEN is discontinued since version 4.4

Applies to RouterOS: v4.3 and below only

Virtualization techonogies enable single physical device to execute multiple different operating systems. Virtualization support in RouterOS allows to run multiple copies of RouterOS sofware and even other supported operating systems. Note that virtualization support depends on system architecture, not all architectures that RouterOS supports allow virtualization.

Ability to run non-RouterOS sofware allows user to run applications that are not included in RouterOS.

Xen is the RouterOS Virtualization system for X86 machines, Xen is based on Xen Virtual machine of Linux.

x86 Virtualization Support

Virtualization support on x86 architecture systems is implemented using Xen hypervisor (http://www.xen.org). This enables RouterOS to run other operating systems that support Xen paravirtualization in "virtual machines" (guests), controlled by RouterOS software (host).

Support for virtualization for x86 architecture systems is included in RouterOS software versions starting with 3.11. To enable virtualization support "xen" package must be installed.

Host RouterOS software sets up virtual machines such that they use file in RouterOS host file system as disk image(s). Additionally host RouterOS can set up virtual ethernet network interfaces between itself and virtual machine. This enables virtual machines to participate in network under control of host RouterOS software.

In order to execute operating system in virtual machine, you need:

- OS kernel that supports Xen paravirtualization
- OS disk image
- (optionally) initial ram disk to use while booting OS in VM

If RouterOS image is used for booting in VM, OS kernel and initial ram disk are not necessary - specifying RouterOS disk image is sufficient. RouterOS images for use by VMs can be created in 2 ways:

- either by taking image from existing RouterOS x86 installation that supports virtualization (version >= 3.11)
- or by using special RouterOS functions to create RouterOS image to use in VM (note that these functions **do not** produce RouterOS image that can be copied and successfully run from physical media!).

The latter approach is more flexible because allows user to specify disk image size.

To be able to run non-RouterOS operating system in VM, you need Linux kernel, disk image and initial ram disk (if necessary) files.

Note that one disk image at the same time can only be used by one VM.

Creating RouterOS image to use in VM

To create RouterOS image to use in VM use "/xen make-routeros-image" command:

```
[admin@MikroTik] /xen> make-routeros-image file-name=ros1.img file-size=32

[admin@MikroTik] /xen> /file print
 # NAME              TYPE            SIZE       CREATION-TIME
 0 ros1.img          .img file       33554432   jun/06/2008 14:47:23
```

This produces 32MB RouterOS image that is ready to use in VM. New RouterOS image is based on host system sofware and therefore contains all sofware packages that are installed on host system, but does not contain host configuration.

Additionally, "make-routeros-image" has "configuration-script" file parameter that can be used to put on initial configuration script in created image. The script will be run on first boot of image.

VM Configuration

All virtualization for x86 architecture related functions are configured under "/xen" menu.

Memory Available to Host RouterOS

By default all the memory is available to host system, for example for system with 1GB of memory:

```
[admin@MikroTik] > /system resource print
                   uptime: 2m4s
                  version: "3.9"
             free-memory: 934116kB
            total-memory: 963780kB
                     cpu: "Intel(R)"
               cpu-count: 2
           cpu-frequency: 2813MHz
                cpu-load: 0
          free-hdd-space: 77728884kB
         total-hdd-space: 79134596kB
  write-sect-since-reboot: 989
        write-sect-total: 989
       architecture-name: "x86"
              board-name: "x86"
[admin@MikroTik] > /xen global-settings print
  memory-for-main: unlimited
```

In some cases this may limit ability to allocate necessary memory for running guest VMs, because host system may have used memory for e.g. filesystem caching purposes. Therefore it is advised to configure limit of memory available to host system (exact value for limit depends on what sofware features are used on host system - in general, the same rules as for choosing amount of physical memory for regular RouterOS installation apply):

```
[admin@MikroTik] > /system resource print
                   uptime: 2m4s
                  version: "3.9"
             free-memory: 934116kB
            total-memory: 963780kB
                     cpu: "Intel(R)"
               cpu-count: 2
           cpu-frequency: 2813MHz
                cpu-load: 0
          free-hdd-space: 77728884kB
         total-hdd-space: 79134596kB
  write-sect-since-reboot: 989
        write-sect-total: 989
```

```
        architecture-name: "x86"
               board-name: "x86"
[admin@MikroTik] > /xen global-settings print
  memory-for-main: unlimited
[admin@MikroTik] > /xen global-settings set memory-for-main=128
[admin@MikroTik] > /system reboot
Reboot, yes? [y/N]:
y
system will reboot shortly

 ....

[admin@MikroTik] > /system resource print
                   uptime: 1m5s
                  version: "3.11"
             free-memory: 114440kB
            total-memory: 131272kB
                     cpu: "Intel(R)"
               cpu-count: 2
            cpu-frequency: 2813MHz
                 cpu-load: 0
           free-hdd-space: 77728884kB
          total-hdd-space: 79134596kB
  write-sect-since-reboot: 794
         write-sect-total: 794
        architecture-name: "x86"
               board-name: "x86"
```

Creating RouterOS VM

Assuming that RouterOS image "ros1.img" is previously made, new VM to run RouterOS can be created:

```
[admin@MikroTik] /xen> add name=ros1 disk-images=hda:ros1.img memory=64 console-telnet-port=64000
[admin@MikroTik] /xen> print detail
Flags: X - disabled, C - configuration-changed
 0 X name="ros1" disk-images="hda:ros1.img" initrd="" kernel="" kernel-cmdline="" cpu-count=1 memory=64 weight=256
     console-telnet-port=64000 state=disabled
```

The following parameters were passed to "add" command:

- disk-images=hda:ros1.img - these parameters specify that file "ros1.img" in host filesystem will be set up as disk "hda" (IDE Primary Master) in guest system;
- memory=64 - this specifies amount of memory for guest VM;
- console-telnet-port=64000 - specifies that host system will listen on port 64000 and once telnetted to, will forward guests console output to telnet client and accept console input from telnet client.

There are few other settings:

- kernel & initrd - VM kernel file to boot and initial ram disk file to use (if specified), as noted before, specifying these is not necessary when booting RouterOS image;
- kernel-cmdline - command line to pass to Linux kernel

- cpu-count - how many CPUs should be made available to VM;
- weight - proportional "importance" of this VM when scheduling multiple VMs for execution. Taking into account that host operating system shares CPUs with all running guest VMs, weight parameter specifies proportional share of CPU(s) that guest operating system will get when multiple operating systems start competing for CPU resource. "Weight" of host operating system is 256. So, for example, if guest VM is also configured with weight 256, if both OSes will be running at 100% CPU usage, both will get equal share of CPU. If guest VM will be configured with weight 128, it will get only 1/3 of CPU.

Starting, Stopping and Connecting to RouterOS VM

To start booting guest VM, enable it:

```
[admin@MikroTik] /xen> enable ros1
[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                                MEMORY     WEIGHT STATE
 0   ros1                                                                64         256    running
```

There are 2 (mutually exclusive, because there is just one virtual console provided for guest VM) ways to connect to console of running VM:

- by using "/xen console <VM name>" command, or
- by using telnet program and connecting to port specified in "console-telnet-port" parameter.

There are multiple ways to stop running VM:

- preferred way is to shut down **from** guest VM (e.g. by connecting to guest VM, logging in and issuing "/system shutdown" command).
- force shutdown from host RouterOS by using "/xen shutdown <VM name>" command;
- simply by disabling VM entry in "/xen" menu, note that this is the most dangerous way of stopping running VM, because guest VM can leave its filesystem in corrupt state (disabling VM entry for VM is the same as unplugging power for physical device).

VM shutdown state can be confirmed in "/xen" menu:

```
[admin@MikroTik] /xen> shutdown ros1


[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                            MEMORY     WEIGHT STATE
 0   ros1                                                            64         256    shutdown
```

In order to boot VM that is shut down, you must either disable and enable VM entry in "/xen" menu or use "/xen start <VM name>" command.

There is also "/xen reboot <VM name>" command, that can be used to restart running guest VM, but it must be taken into account that using this command is dagerous - although it instructs guest VM to reboot, in most cases it does not cause guest to flush its filesystem and terminate correctly.

If any guest VM related settings are changed for VM entry in "/xen" menu, if guest VM is running, those settings are not applied immediately (because that would involve destroying VM and starting it again). Instead, VM is marked as "configuration-changed" and new settings will be applied on next reboot. For example:

```
[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
```

```
 #   NAME                                                                    MEMORY     WEIGHT STATE
 0   ros1                                                                    64         256    running
[admin@MikroTik] /xen> set ros1 memory=32
[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                                    MEMORY     WEIGHT STATE
 0 C ros1                                                                    32         256    running
[admin@MikroTik] /xen> shutdown ros1

[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                                    MEMORY     WEIGHT STATE
 0   ros1                                                                    32         256    shutdown
[admin@MikroTik] /xen> start ros1

[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                                    MEMORY     WEIGHT STATE
 0   ros1                                                                    32         256    running
```

After this command sequence memory of running guest is actually 32Mb.

Reconfiguring RouterOS VM Image

With "/xen reconfigure-routeros-image", RouterOS configuration from existing RouterOS image can be wiped out and new configuration script put on (the script will be executed when VM using this image will next get started):

```
[admin@MikroTik] /xen> reconfigure-routeros-image file-name=ros1.img configuration-script=script.file
```

Configuring VM Networking

In order for guest VM to participate in network, virtual interfaces that connect guest VM with host must be created. Virtual network connection with guest VM can be thought of as point-to-point ethernet network connection, which terminates in guest VM as "/interface ethernet" type interface and in host as "/interface virtual-ethernet" interface. By configuring appropriate data forwarding (either by bridging or routing) to/from virtual-ethernet interface in host system, guest VM can be allowed to participate in real network.

Configuring Network Interfaces for Guest VM

Network interfaces that will appear in guest VM as ethernet interfaces are configured in "/xen interface" menu:

```
[admin@MikroTik] /xen interface> add virtual-machine=ros1 type=dynamic
[admin@MikroTik] /xen interface> print detail
Flags: X - disabled, A - active
 0   virtual-machine=ros1 vm-mac-addr=02:1C:AE:C1:B4:B2 type=dynamic static-interface=none
    dynamic-mac-addr=02:38:19:0C:F3:98 dynamic-bridge=none
```

Above command creates interface for guest VM "ros1" with type "dynamic".

There are 2 types of interfaces:

- dynamic - endpoint of virtual network connection in host ("/interface virtual-ethernet") will be created dynamically when guest VM will be booted. By using this type of interface user avoids manually creating

endpoint of virtual connection in host, at the expense of limited flexibility how this connection can be used (e.g. there is no way how to reliably assign IP address to dynamically created interface). Currently, it can only be automatically added to bridge specified in "dynamic-bridge" parameter. This behaviour is similar to dynamic WDS interfaces for wireless WDS links.

- static - endpoint of virtual network connection in host ("/interface virtual-ethernet") must be manually created. This type of interface allows maximum flexibility because interface that will connect with guest VM is previously known (therefore IP addresses can be added, interface can be used in filter rules, etc.), at the expense of having to create "/interface virtual-ethernet" manually.

VM interfaces have the following parameters:

- virtual-machine - to which VM this interface belongs;
- vm-mac-addr - MAC address of ethernet interface in **guest** system;
- type - interface type as described above
- static-interface - when "type=static", this parameter specifies which "/interface virtual-ethernet" in host system will be connected with guest;
- dynamic-mac-addr - when "type=dynamic", automatically created "/interface virtual-ethernet" in **host** system will have this MAC address;
- dynamic-bridge - when "type=dynamic", dynamically created "/interface virtual-ethernet" will automatically get added as bridge port to this bridge.

Configuring Dynamic Interfaces

To create virtual connection that will have its endpoint in host dynamically made, use the following command:

```
[admin@MikroTik] /xen interface> add virtual-machine=ros1 type=dynamic

[admin@MikroTik] /xen interface> print detail

Flags: X - disabled, A - active

 0   virtual-machine=ros1 vm-mac-addr=02:1C:AE:C1:B4:B2 type=dynamic static-interface=none dynamic-mac-addr=02:38:19:0C:F3:98 dynamic-bridge=none
```

After enabling "ros1" VM, you can confirm that new virtual-ethernet interface is made with given dynamic-mac-addr:

```
[admin@MikroTik] /xen> /interface virtual-ethernet print

Flags: X - disabled, R - running

 #    NAME                                                                                                MTU   ARP        MAC-ADDRESS

 0  R vif1                                                                                                1500  enabled    02:38:19:0C:F3:98
```

And in guest VM ethernet interface is available with given vm-mac-addr:

```
[admin@Guest] > int ethernet print

Flags: X - disabled, R - running, S - slave

 #    NAME                                                                                                MTU   MAC-ADDRESS       ARP

 0 R  ether1                                                                                              1500  02:1C:AE:C1:B4:B2 enabled
```

By configuring "dynamic-bridge" setting, virtual-ethernet interface can be automatically added as bridge port to some bridge in host system. For example, if it is necessary to forward traffic between "ether1" interface on host and VM "ros1" ethernet interface, the following steps must be taken:

Create bridge on host system and add "ether1" as bridge port:

```
[admin@MikroTik] > /interface bridge add name=to-ros1
[admin@MikroTik] > /interface bridge port add bridge=to-ros1 interface=ether1
```

Next, specify that virtual-ethernet should automatically get added as bridge port:

```
[admin@MikroTik] /xen interface> print detail
Flags: X - disabled, A - active
 0 A virtual-machine=ros1 vm-mac-addr=02:1C:AE:C1:B4:B2 type=dynamic static-interface=none dynamic-mac-addr=02:38:19:0C:F3:98 dynamic-bridge=none
[admin@MikroTik] /xen interface> set 0 dynamic-bridge=to-ros1
```

After this virtual-ethernet interface is added as bridge port on host:

```
[admin@MikroTik] /xen interface> /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE                                  BRIDGE                                     PRIORITY PATH-COST  HORIZON
 0    ether1                                     to-ros1                                    0x80     10         none
 1  D vif1                                       to-ros1                                    0x80     10         none
```

By using similar configuration, user can, for example, "pipe" all traffic through guest VM - if there are 2 physical interfaces in host, user can create 2 bridges and bridge all traffic through guest VM (assuming that operating system in guest is configured in such a way that ensures data forwarding between its interfaces).

Configuring Static Interfaces

To create virtual connection whose endpoint in host system will be static interface, at first create static virtual-ethernet interface:

```
[admin@MikroTik] /interface virtual-ethernet> add name=static-to-ros1 disabled=no
[admin@MikroTik] /interface virtual-ethernet> print
Flags: X - disabled, R - running
 #    NAME                                                                                MTU   ARP        MAC-ADDRESS
 0  R vif1                                                                                1500  enabled    02:38:19:0C:F3:98
 1    static-to-ros1                                                                      1500  enabled    02:3A:1B:DB:FC:CF
```

Next, create interface for guest VM:

```
[admin@MikroTik] /xen interface> add virtual-machine=ros1 type=static static-interface=static-to-ros1
[admin@MikroTik] /xen interface> print
Flags: X - disabled, A - active
 #   VIRTUAL-MACHINE                                                                      TYPE    VM-MAC-ADDR
 0 A ros1                                                                                 dynamic 02:1C:AE:C1:B4:B2
 1 A ros1                                                                                 static  02:DF:66:CD:E9:74
```

Now we can confirm that virtual-ethernet interface is active:

```
[admin@MikroTik] /xen interface> /interface virtual-ethernet print
Flags: X - disabled, R - running
 #    NAME                                                                                MTU   ARP        MAC-ADDRESS
 0  R static-to-ros1                                                                      1500  enabled    02:3A:1B:DB:FC:CF
 1  R vif1                                                                                1500  enabled    02:38:19:0C:F3:98
```

And in guest system:

```
[admin@Guest] > /interface ethernet print
Flags: X - disabled, R - running, S - slave
 #    NAME                                                                                MTU   MAC-ADDRESS       ARP
 0 R  ether1                                                                              1500  02:1C:AE:C1:B4:B2 enabled
```

```
 1 R  ether2                                                                1500  02:DF:66:CD:E9:74 enabled
```

Having static interface in host system allows to use interface in configuration wherever specifying interface is necessary, e.g. adding ip address:

```
[admin@MikroTik] > ip address add interface=static-to-ros1 address=1.1.1.1/24
```

In similar way we add IP address to appropriate interface in guest system and confirm that routing is working:

```
[admin@Guest] > /ip address add interface=ether2 address=1.1.1.2/24
[admin@Guest] > /ping 1.1.1.1
1.1.1.1 64 byte ping: ttl=64 time=5 ms
1.1.1.1 64 byte ping: ttl=64 time<1 ms
1.1.1.1 64 byte ping: ttl=64 time<1 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0/1.6/5 ms
```

Running non-RouterOS Systems as Guests

Xen hypervisor based virtualization for x86 architectures that is used in RouterOS allow to run other operating systems that use Linux kernel that has Xen guest support (DomU support in Xen terminology).

In order to run non-RouterOS system as guest on RouterOS host, you need:

- OS image file
- OS Linux kernel file
- optionally initial ram disk file

There are several ways how to prepare necessary files:

- using already prepared, ready to boot images (the easiest way);
- installing operating system along with necessary virtualization packages and using image from installed system (medium complexity);
- installing operating system, taking image, recompiling Linux kernel, adjusting system to boot under Xen hypervisor (the most complex).

Using Ready to Boot Image

If you have operational Debian GNU/Linux based system with Xen installed, you can use xen-tools scripts (http://www. xen-tools. org) to create/install images and use Xen kernel and initrd from your distribution. Another opportunity is to use already prepared images that are available for download, for example from http://jailtime.org. Note that images **do not** contain the kernel itself, therefore these images can only be used after taking appropriate kernel and initrd file from real distribution.

Additionally, here we provide some sets of files ready to use in guest VMs.

ClarkConnect 4.2 Community Edition SP1 Image

Image is prepared from installation ISO (ftp://ftp.clarkconnect.com/4.2/iso/community-4.2.SP1.iso) and additional Xen kernel package (ftp://ftp.clarkconnect.com/4.2/other/kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm).Minimum software is installed.

Archive contains the following files:

- kernel: vmlinuz-2.6.18-8.1.14.3.ccxen
- initial RAM disk: clark.initrd.rgz, clark.otherinitrd.rgz (either one can be used)
- disk image: clark.img

To use this image in guest VM under RouterOS (remember that you have to upload files from archive to your RouterOS host), use the following command:

```
[admin@MikroTik] /xen> add disk=hda disk-image=clark.img
initrd=clark.otherinitrd.rgz kernel=vmlinuz-2.6.18-8.1.14.3.ccxen
kernel-cmdline="root=/dev/hda1" memory=128 name=clark
```

Password for *root* user is *rootroot*.

Archive can be dowloaded here: http://www.mikrotik.com/download/clark.tar.bz2

CentOS 5.1 Image

Image is prepared using netinstall ISO (ISO file: CentOS-5.1-i386-netinstall.iso, available from mirrors listed at http://isoredirect.centos.org/centos/5/isos/i386/) and network based installation. Minimum software is installed.

Archive contains the following files:

- kernel: vmlinuz-2.6.18-53.el5xen
- inital RAM disk: centos.initrd.rgz
- disk image: centos.img

To use this image in guest VM under RouterOS (remember that you have to upload files from archive to your RouterOS host), use the following command:

```
[admin@MikroTik] /xen> add disk=hda disk-image=centos.img
initrd=centos.initrd.rgz kernel=vmlinuz-2.6.18-53.el5xen
kernel-cmdline="ro root=/dev/hda1" memory=512 name=centos
```

Password for *root* user is *rootroot*.

Archive can be dowloaded here: http://www.mikrotik.com/download/centos.tar.bz2

Installing OS with Virtualization Support

One of ways to simplify OS installation is to install it in image file using some full virtualization software, like VMWare or QEMU. This allows to produce ready to use image file and does not require any additional hardware.

Example: Preparing ClarkConnect Community Edition 4.2 SP1 Image

Below find instructions on how to get ClarkConnect 4.2 Community Edition run as guest VM. Note that ClarkConnect installation does not provide support for virtualization by default, therefore additional tweaks will be necessary.

Installing ClarkConnect

At first, create image where ClarkConnect will be installed:

```
~/xen$ qemu-img create clark.img 1Gb
Formatting 'clark.img', fmt=raw, size=1048576 kB
```

Next, start installation from ClarkConnect installation ISO image:

```
~/xen$ sudo qemu -hda clark.img -cdrom community-4.2.SP1.iso -net nic,vlan=0,macaddr=00:01:02:03:04:aa -net tap,vlan=0,ifname=tap0 -m 128 -boot d
```

Proceed with installation, creating one root partition and (optionally) swap space. Take into account disk size when selecting software packages to install. In this example disk is partitioned with 800MB root partition size and the rest of image for swap. Note that QEMU is instructed to emulate ethernet card, during installation this card is configured with IP address 10.0.0.23/24.

ClarkConnect installation does not provide support for virtualization by default, therefore virtualization support will have to be added manually. ClarkConnect distributes Xen-aware kernel package separately from installation, available at: ftp://ftp.clarkconnect.com/4.2/other/kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm

In order to install this package we have to put it on newly created image. To do this, boot new image:

```
~/xen$ sudo qemu -hda clark.img -net nic,vlan=0,macaddr=00:01:02:03:04:aa -net tap,vlan=0,ifname=tap0 -m 128
```

Assuming that networking with QEMU virtual machine is configured properly, we can use SCP to put on package file:

```
~/xen$ scp ./kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm root@10.0.0.23:/

The authenticity of host '10.0.0.23 (10.0.0.23)' can't be established.

RSA key fingerprint is 70:84:b8:c5:6d:62:37:d1:1e:96:29:d0:77:46:6a:0c.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '10.0.0.23' (RSA) to the list of known hosts.

root@10.0.0.23's password:

kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm                                                                100%   16MB   2.0MB/s   00:08
```

Next, connect to ClarkConnect and install kernel package. Note that this package is not entirely compatible with ClarkConnect 4.2 SP1 system and proper installation fails, but taking into account that the only purpose of installing this package is to get Xen enabled kernel and drivers, forced installation is fine, except that module dependency file must be created manually:

```
~/xen$ ssh root@10.0.0.23
root@10.0.0.23's password:
Last login: Tue Jun 10 07:20:34 2008
[root@server ~]# cd /
[root@server /]# rpm -i kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm --force
```

```
--nodeps
Usage: new-kernel-pkg [-v] [--mkinitrd] [--rminitrd]
       [--initrdfile=<initrd-image>] [--depmod] [--rmmoddep]
       [--kernel-args=<args>] [--banner=<banner>]
       [--make-default] <--install | --remove>
<kernel-version>
       (ex: new-kernel-pkg --mkinitrd --depmod --install 2.4.7-2)
error: %post(kernel-xen-2.6.18-8.1.14.3.cc.i686) scriptlet failed, exit
 status 1
[root@server /]# ls /boot
config-2.6.18-53.1.13.2.cc    initrd-2.6.18-53.1.13.2.cc.img
symvers-2.6.18-8.1.14.3.ccxen.gz  vmlinuz-2.6.18-53.1.13.2.cc
xen-syms-2.6.18-8.1.14.3.cc
config-2.6.18-8.1.14.3.ccxen  memtest86+-1.26
System.map-2.6.18-53.1.13.2.cc    vmlinuz-2.6.18-8.1.14.3.ccxen
grub                          symvers-2.6.18-53.1.13.2.cc.gz
System.map-2.6.18-8.1.14.3.ccxen  xen.gz-2.6.18-8.1.14.3.cc
[root@server /]# depmod -v 2.6.18-8.1.14.3.ccxen -F
/boot/System.map-2.6.18-8.1.14.3.ccxen
 ....
```

Next, copy out some files from installed system:

- Xen enabled kernel (/boot/vmlinuz-2.6.18-8.1.14.3.ccxen)
- initial ram disk (/boot/initrd-2.6.18-53.1.13.2.cc.img)

```
~/xen$ scp root@10.0.0.23:/boot/vmlinuz-2.6.18-8.1.14.3.ccxen ./

root@10.0.0.23's password:

vmlinuz-2.6.18-8.1.14.3.ccxen                                                 100% 2049KB   2.0MB/s   00:01

~/xen$ scp root@10.0.0.23:/boot/initrd-2.6.18-53.1.13.2.cc.img ./

root@10.0.0.23's password:

initrd-2.6.18-53.1.13.2.cc.img                                                100%  434KB 433.8KB/s   00:00
```

Default ClarkConnect installation does not execute login process on Xen virtual console, so in order to have login available on virtual console accessible from RouterOS with "/xen console <VM name>" command, virtual console device should get made inside image (mknod /dev/xvc0 c 204 191).

Default ClarkConnect initial ram disk does not support booting from Xen virtual disk because it does not contain driver for virtual disk. To overcome this problem initial ram disk must be updated.

Updating initrd Manually

One opportunity to make initial ram disk that would support booting from virtual disk is to manually put virtual disk driver in initrd and update it to load this module.

At first we extract contents of initial ram disk that was copied from ClarkConnect image:

```
~/xen$ file initrd-2.6.18-53.1.13.2.cc.img

initrd-2.6.18-53.1.13.2.cc.img: gzip compressed data, from Unix, last modified: Tue Jun 10 14:01:27 2008, max compression

~/xen$ mv initrd-2.6.18-53.1.13.2.cc.img clarkinitrd.gz

~/xen$ gunzip clarkinitrd.gz

~/xen$ file clarkinitrd

clarkinitrd: ASCII cpio archive (SVR4 with no CRC)
```

```
~/xen$ mkdir initrd
~/xen$ cd initrd/
~/xen/initrd$ sudo cpio -idv --no-absolute-filenames < ../clarkinitrd
.
etc
bin
bin/insmod
bin/nash
bin/modprobe
sysroot
sys
lib
lib/sd_mod.ko
lib/libata.ko
lib/scsi_mod.ko
lib/ata_piix.ko
lib/ext3.ko
lib/jbd.ko
sbin
dev
dev/console
dev/systty
dev/tty3
dev/tty2
dev/tty4
dev/ram
dev/tty1
dev/null
init
loopfs
proc
1990 blocks
~/xen/initrd$ cat init
#!/bin/nash

mount -t proc /proc /proc
setquiet
echo Mounted /proc filesystem
echo Mounting sysfs
mount -t sysfs none /sys
echo "Loading scsi_mod.ko module"
insmod /lib/scsi_mod.ko
echo "Loading sd_mod.ko module"
insmod /lib/sd_mod.ko
echo "Loading libata.ko module"
insmod /lib/libata.ko
echo "Loading ata_piix.ko module"
```

```
insmod /lib/ata_piix.ko
echo "Loading jbd.ko module"
insmod /lib/jbd.ko
echo "Loading ext3.ko module"
insmod /lib/ext3.ko
echo Creating block devices
mkdevices /dev
echo Creating root device
mkrootdev /dev/root
umount /sys
echo Mounting root filesystem
mount -o defaults --ro -t ext3 /dev/root /sysroot
echo Switching to new root
switchroot /sysroot
```

From the above we see that init script in initrd image loads drivers for SCSI and ATA disks, as well as EXT3 filesystem modules. In order for ClarkConnect to boot under Xen we have to replace hardware drivers with Xen virtual disk driver and EXT3 filesystem modules with appropriate modules for Xen kernel. Take these modules from installed ClarkConnect system:

```
~/xen/initrd$ cd lib/
~/xen/initrd/lib$ ls
ata_piix.ko  ext3.ko  jbd.ko  libata.ko  scsi_mod.ko  sd_mod.ko
~/xen/initrd/lib$ sudo rm ata_piix.ko libata.ko scsi_mod.ko sd_mod.k
~/xen/initrd/lib$ scp
root@10.0.0.23:/lib/modules/2.6.18-8.1.14.3.ccxen/kernel/fs/jbd/jbd.ko
./
root@10.0.0.23's password:
jbd.ko

100%   70KB  69.8KB/s
00:00
~/xen/initrd/lib$ sudo scp
root@10.0.0.23:/lib/modules/2.6.18-8.1.14.3.ccxen/kernel/fs/ext3/ext3.ko
 ./
root@10.0.0.23's password:
ext3.ko

100%  141KB 141.5KB/s
00:00
~/xen/initrd/lib$ sudo scp
root@10.0.0.23:/lib/modules/2.6.18-8.1.14.3.ccxen/kernel/drivers/xen/blkfront/xenblk.ko
 ./
root@10.0.0.23's password:
xenblk.ko

100%   22KB  22.0KB/s
00:00
```

Next, update init script so that it loads Xen virtual disk driver. Final init script should look like this:

```
~/xen/initrd$ cat init
#!/bin/nash

mount -t proc /proc /proc
setquiet
echo Mounted /proc filesystem
echo Mounting sysfs
mount -t sysfs none /sys
echo "Loading xenblk.ko module"
insmod /lib/xenblk.ko
echo "Loading jbd.ko module"
insmod /lib/jbd.ko
echo "Loading ext3.ko module"
insmod /lib/ext3.ko
echo Creating block devices
mkdevices /dev
echo Creating root device
mkrootdev /dev/root
umount /sys
echo Mounting root filesystem
mount -o defaults --ro -t ext3 /dev/root /sysroot
echo Switching to new root
switchroot /sysroot
```

Create initrd file from directory structure with modifications that have been made:

```
~/xen/initrd$ find * | cpio -o -H newc -O ../clarkinitrd.new
~/xen/initrd$ find . -depth | cpio -ov --format=newc > ../clark.initrd
~/xen/initrd$ cd ../
~/xen$ gzip -c -9 < clark.initrd > clark.initrd.rgz
```

Using mkinitrd Utility

Instead of creating initial ram disk manually as described above is possible to use *mkinitrd* utility available in ClarkConnect distribution. After Xen kernel package is installed as shown above, initial ram disk can be created with command (note that this command must be executed in ClarkConnect, e.g. running in QEMU VM):

```
[root@server /]# mkinitrd clark.otherinitrd.rgz 2.6.18-8.1.14.3.ccxen --omit-scsi-modules --omit-raid-modules --omit-lvm-modules --with=xenblk
```

After this, newly created clark.otherinitrd.rgz must be copied from ClarkConnect image.

Adding ClarkConnect VM in RouterOS

Finally upload files (dont forget to shut down QEMU that executes image) to host RouterOS and create guest VM entry:

```
[admin@MikroTik] /xen> print detail
Flags: X - disabled
 1 X name="clark" disk=hda disk-image="clark.img" initrd="clark.otherinitrd.rgz" kernel="vmlinuz-2.6.18-8.1.14.3.ccxen"
     kernel-cmdline="root=/dev/hda1" cpu-count=1 memory=128 weight=256 console-telnet-port=64000 state=disabled
```

Note that VM is configured with files that were made in previous steps. Also pay attention to "kernel-cmdline" parameter that is supplied. This instructs ClarkConnect where its root file system is - as we are providing ClarkConnect image with "disk=hda", and during installation root filesystem was made as first partition in image, root file system is on device /dev/hda1.

On first boot of ClarkConnect, it will detect changes in hardware and also enable login on virtual console device.

Example: Preparing Centos 5.1 Image

CentOS ir RedHat Linux based Linux distribution. Distribution includes necessary software packages for virtualization support, therefore installing CentOS image that supports virtualization is rather simple.

Installing CentOS 5.1

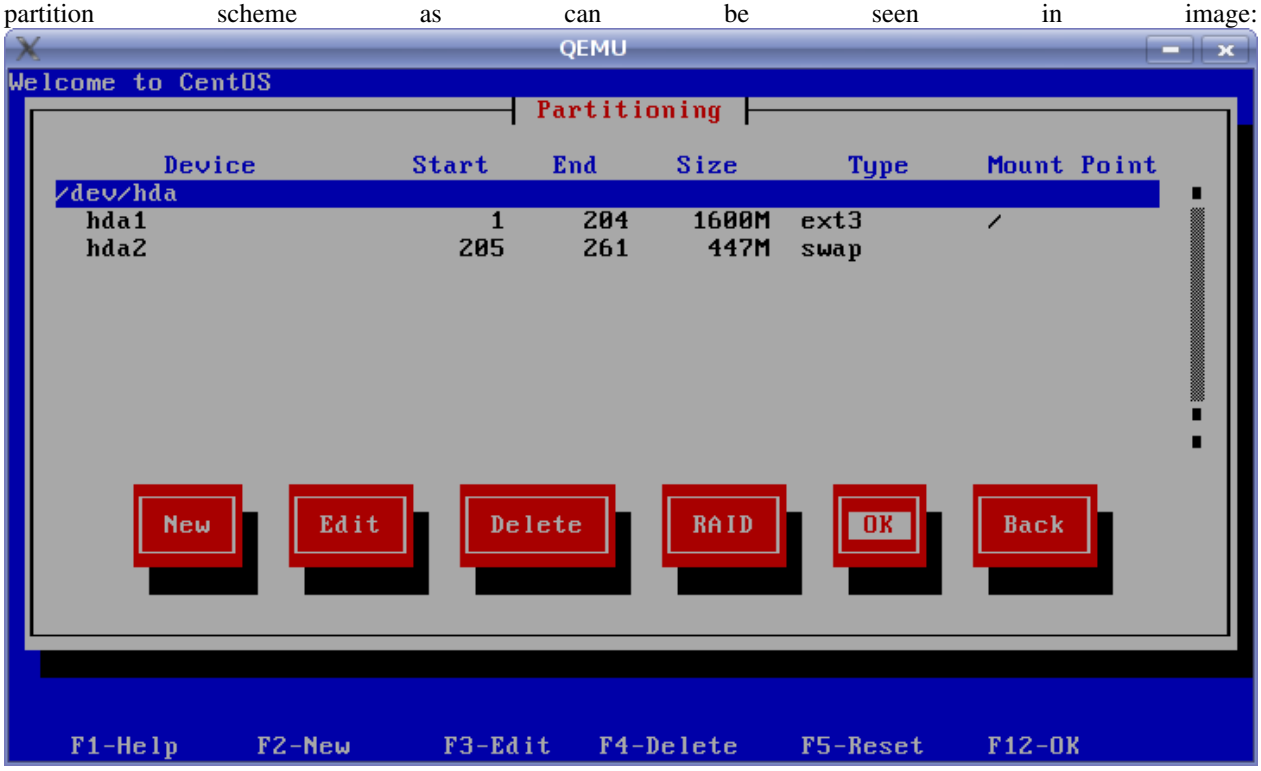
To create example CentOS image, we use QEMU for CentOS installation the same way as in previous ClarkConnect example.

Create image file and start QEMU with CentOS netinstall ISO image:

```
~/xen$ qemu-img create centos.img 2Gb
Formatting 'centos.img', fmt=raw, size=2097152 kB
~/xen$ sudo qemu -hda centos.img -cdrom=CentOS-5.1-i386-netinstall.iso
-net nic,vlan=0,macaddr=00:01:02:03:04:aa -net tap,vlan=0,ifname=tap0
-m 256
```

Note that netinstall ISO image is used - sofware packages will be downloaded from network. This means that network connectivity of QEMU VM must be configured and running.

During installation follow partition scheme as in previous example for ClarkConnect. Example image is created with partition scheme as can be seen in image:



Also during installation select "Virtualization" sofware set:

QEMU
Welcome to CentOS
Package selection
The default installation of CentOS includes a set of software
applicable for general internet usage. What additional tasks
would you like your system to include support for?
[ ] Server
[ ] Server - GUI
[*] Virtualization
[ ] Clustering
[*] Customize software selection
OK
Back
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

When installation is complete, CentOS image does not boot under QEMU emulator because it does not support running Xen hypervisor. Nevertheless this does not matter, because all necessary sofware for running as guest is already installed in image. Still this forces to take different approach for extracting necessary files from image (for ClarkConnect this got done by connecting to VM running under QEMU and copying files out).

Preparing Initial Ram Disk

To take Xen kernel from CentOS image and to prepare initrd (that would include driver for virtual disk), use the following steps.

Mount root partition of image using loopback device (note that 1st partition in image starts with sector 63 therefore we use offset in image file to point to beginning of partition:

```
# mount centos.img /mnt -o loop,offset=$[512*63]
```

Next, copy out kernel file:

```
# cp /mnt/boot/vmlinuz-2.6.18-53.el5xen ./
```

To prepear initrd file we use mkinitrd tool. To force it to work on mounted image, use chroot command:

```
# chroot /mnt /bin/sh

sh-3.1# mkinitrd centos.initrd.rgz 2.6.18-53.el5xen --omit-scsi-modules --omit-raid-modules --omit-lvm-modules --with=xenblk

sh-3.1# exit
```

Adding CentOS VM in RouterOS

Now files to be used for running guest VM (kernel, newly made initrd and image) have to be put on RouterOS and appropriate VM entry should be made:

```
[admin@MikroTik] /xen> print detail

Flags: X - disabled

 0 X name="centos" disk=hda disk-image="centos.img" initrd="centos.initrd.rgz" kernel="vmlinuz-2.6.18-53.el5xen"

     kernel-cmdline="ro root=/dev/hda1" cpu-count=1 memory=256 weight=256 console-telnet-port=64000 state=disabled
```

Notice that CentOS kernel is also passed arguments of which partition should be used for root file system, similar to ClarkConnect.

Adding Virtualization Support to Your Favourite Linux Based OS

To be able to run your favourite OS/distribution in guest VM, it must support Xen DomU, therefore enabling Xen support most likely will involve recompiling kernel. Disk and virtual network interface devices have to be accessed by Xen *netfront* and *blockfront* drivers, therefore you should make sure that these drivers are included in your system, either directly in kernel or as modules. Kernel must be compiled with PAE support.

Depending on Linux kernel that your distribution uses, it is possible that kernel source does not have support for Xen. This may mean that patching of kernel is necessary. In such cases you can refer to distributions that use similar kernel version and have vendor patches for Xen support.

... Some time later maybe example will come...

Manual:Interface/Virtual-ethernet

Applies to RouterOS: v4.x

Summary

To connect your virtual routers to RouterOS host system you either have to assign interface for your guest (possible only on MetaROUTER) or you can add virtual Ethernet interface that is described in this document.

May contain either static or dynamic interface. Static interfaces should be configured in virtual-ethernet menu and then assigned to virtual machine in **/kvm interface** (for KVM) or **/metarouter interface** (for MetaROUTER). Dynamic interfaces will be recreated automatically on each reboot and will contain new MAC address.

Note: Virtual-ethernets will be automatically removed even if configured as static in /kvm interface menu

Requirements

This menu becomes available:

- on *x86* architeecture you have to have *kvm* packge installed
- on *mipsbe* architecture RouterBOARDS
- on *ppc* architecture RouterBOARDS, except RB333, RB600 and variants.

Virtual Ethernet creation

Menu: */interface virtual-ethernet add*

Property	Desciption
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only, default: enabled*)	ARP protocol resolution mode: • *disabled* - interface is not replying to ARP requests • *enabled* - interface is replaying to all ARP requests on its MAC address • *proxy-arp* - interface is replying to all ARP requests even if it is not interface MAC address • *reply-only - interface replies only to known (static entries in ARP table) sources*
comment (*text*)	Short description of the item
copy-from (*number*)	Item number to copy settings from to create new item
disabled (*yes* | *no default: yes*)	identifies if entry is part of active coniguration
mac-address (*MAC address default: automatically generated*)	MAC address of interface. If automatically generated, then this pattern will be used *02:XX:XX:XX:XX:XX*
mtu (*0 65536 default: 1500*)	maximal transmission unit of the interface
name (*text default: tapX or vifX*)	Interface name where, if auto-generated, X is inreased if previous valid number already exists, starts with 1. tap is on x86 vif is on RouterBOARD platform.

See Also

- KVM
- MetaROUTER

Article Sources and Contributors

Manual:MPLS *Source*: http://wiki.mikrotik.com/index.php?oldid=23554 *Contributors*: Eising, Marisb, Mplsguy, Normis, Route, SergejsB

Manual:MPLS/Overview *Source*: http://wiki.mikrotik.com/index.php?oldid=23235 *Contributors*: Marisb, Mplsguy, Normis, Route

Manual:MPLS over PPPoE *Source*: http://wiki.mikrotik.com/index.php?oldid=16772 *Contributors*: Marisb

Manual:MPLS/EXP bit behaviour *Source*: http://wiki.mikrotik.com/index.php?oldid=16502 *Contributors*: Marisb, Mplsguy, Normis

Manual:MPLS/LDP *Source*: http://wiki.mikrotik.com/index.php?oldid=24648 *Contributors*: Marisb

Manual:MPLSVPLS *Source*: http://wiki.mikrotik.com/index.php?oldid=16498 *Contributors*: Eep, Mag, Marisb, Mplsguy, Normis

Manual:Cisco VPLS *Source*: http://wiki.mikrotik.com/index.php?oldid=16871 *Contributors*: Janisk, Marisb, Mplsguy, Normis

Manual:BGP based VPLS *Source*: http://wiki.mikrotik.com/index.php?oldid=22990 *Contributors*: Eep, Janisk, Marisb, Mplsguy, Normis

Manual:VPLS Control Word *Source*: http://wiki.mikrotik.com/index.php?oldid=25090 *Contributors*: Marisb

Manual:Virtual Routing and Forwarding *Source*: http://wiki.mikrotik.com/index.php?oldid=16975 *Contributors*: Eep, Janisk, Marisb, Normis, Route

Manual:Layer-3 MPLS VPN example *Source*: http://wiki.mikrotik.com/index.php?oldid=16990 *Contributors*: Janisk, Marisb, Normis, Route

Manual:EBGP as PE-CE routing protocol *Source*: http://wiki.mikrotik.com/index.php?oldid=23196 *Contributors*: Marisb, Route

Manual:OSPF as PE-CE routing protocol *Source*: http://wiki.mikrotik.com/index.php?oldid=16913 *Contributors*: Janisk, Marisb, Route

Manual:TE Tunnels *Source*: http://wiki.mikrotik.com/index.php?oldid=16522 *Contributors*: Marisb, Mplsguy, Normis

Manual:TE tunnel auto bandwidth *Source*: http://wiki.mikrotik.com/index.php?oldid=16517 *Contributors*: Marisb, Mplsguy

Manual:MPLS/Traffic-eng *Source*: http://wiki.mikrotik.com/index.php?oldid=17239 *Contributors*: Marisb

Manual:Interface/Traffic Engineering *Source*: http://wiki.mikrotik.com/index.php?oldid=22126 *Contributors*: Janisk, Marisb

Manual:Router AAA *Source*: http://wiki.mikrotik.com/index.php?oldid=25805 *Contributors*: Janisk, Marisb, Normis

Manual:PPP AAA *Source*: http://wiki.mikrotik.com/index.php?oldid=22902 *Contributors*: Janisk, Marisb

Manual:RADIUS Client *Source*: http://wiki.mikrotik.com/index.php?oldid=22741 *Contributors*: Agris, Janisk, Marisb, Normis, SergejsB, Uldis

Manual:User Manager *Source*: http://wiki.mikrotik.com/index.php?oldid=19155 *Contributors*: Akangage, Bhhenry, Binhtanngo2003, Cmit, Comnetisp, Eep, Girts, Hellbound, Janisk, Levipatick, Marisb, Nest, Normis, Polokus, Rtkrh10, SergejsB, Uldis

Manual:Hotspot Introduction *Source*: http://wiki.mikrotik.com/index.php?oldid=25314 *Contributors*: Marisb

Manual:Customizing Hotspot *Source*: http://wiki.mikrotik.com/index.php?oldid=25437 *Contributors*: Marisb, SergejsB

Manual:IP/Hotspot *Source*: http://wiki.mikrotik.com/index.php?oldid=19414 *Contributors*: Janisk, Marisb, Normis, SergejsB, Vitell

Manual:Virtualization *Source*: http://wiki.mikrotik.com/index.php?oldid=20250 *Contributors*: Danielillu, Janisk, Marisb, Normis

Manual:KVM *Source*: http://wiki.mikrotik.com/index.php?oldid=24077 *Contributors*: Janisk, Marisb, Megis, Normis, Route

Manual:Xen *Source*: http://wiki.mikrotik.com/index.php?oldid=16862 *Contributors*: Janisk, Marisb, Mplsguy, Normis, SergejsB

Manual:Interface/Virtual-ethernet *Source*: http://wiki.mikrotik.com/index.php?oldid=22659 *Contributors*: Janisk, Marisb, Normis

Image Sources, Licenses and Contributors

Image:Icon-note.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Icon-note.png *License*: unknown *Contributors*: Marisb, Route
Image:Version.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Version.png *License*: unknown *Contributors*: Normis
Image:mpls-pppoe-f.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Mpls-pppoe-f.png *License*: unknown *Contributors*: Marisb
Image:mpls-pppoe-vpls.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Mpls-pppoe-vpls.png *License*: unknown *Contributors*: Marisb
Image:VPLS.png *Source*: http://wiki.mikrotik.com/index.php?title=File:VPLS.png *License*: unknown *Contributors*: Karliskarlis
File:VPLS_CW.png *Source*: http://wiki.mikrotik.com/index.php?title=File:VPLS_CW.png *License*: unknown *Contributors*: Marisb
Image:l3vpn-simple.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L3vpn-simple.png *License*: unknown *Contributors*: Route
Image:l3vpn-two-customers.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L3vpn-two-customers.png *License*: unknown *Contributors*: Route
Image:l3vpn.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L3vpn.png *License*: unknown *Contributors*: Route
File:bgp_pe_ce.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Bgp_pe_ce.png *License*: unknown *Contributors*: Route
Image:ospf-pe-ce.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-pe-ce.png *License*: unknown *Contributors*: Route
Image:ospf-pe-ce-sham-links.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-pe-ce-sham-links.png *License*: unknown *Contributors*: Route
Image:Icon-warn.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Icon-warn.png *License*: unknown *Contributors*: Marisb, Route
Image:Centos_partitioning.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Centos_partitioning.png *License*: unknown *Contributors*: Mplsguy
Image:Centos_packages.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Centos_packages.png *License*: unknown *Contributors*: Mplsguy